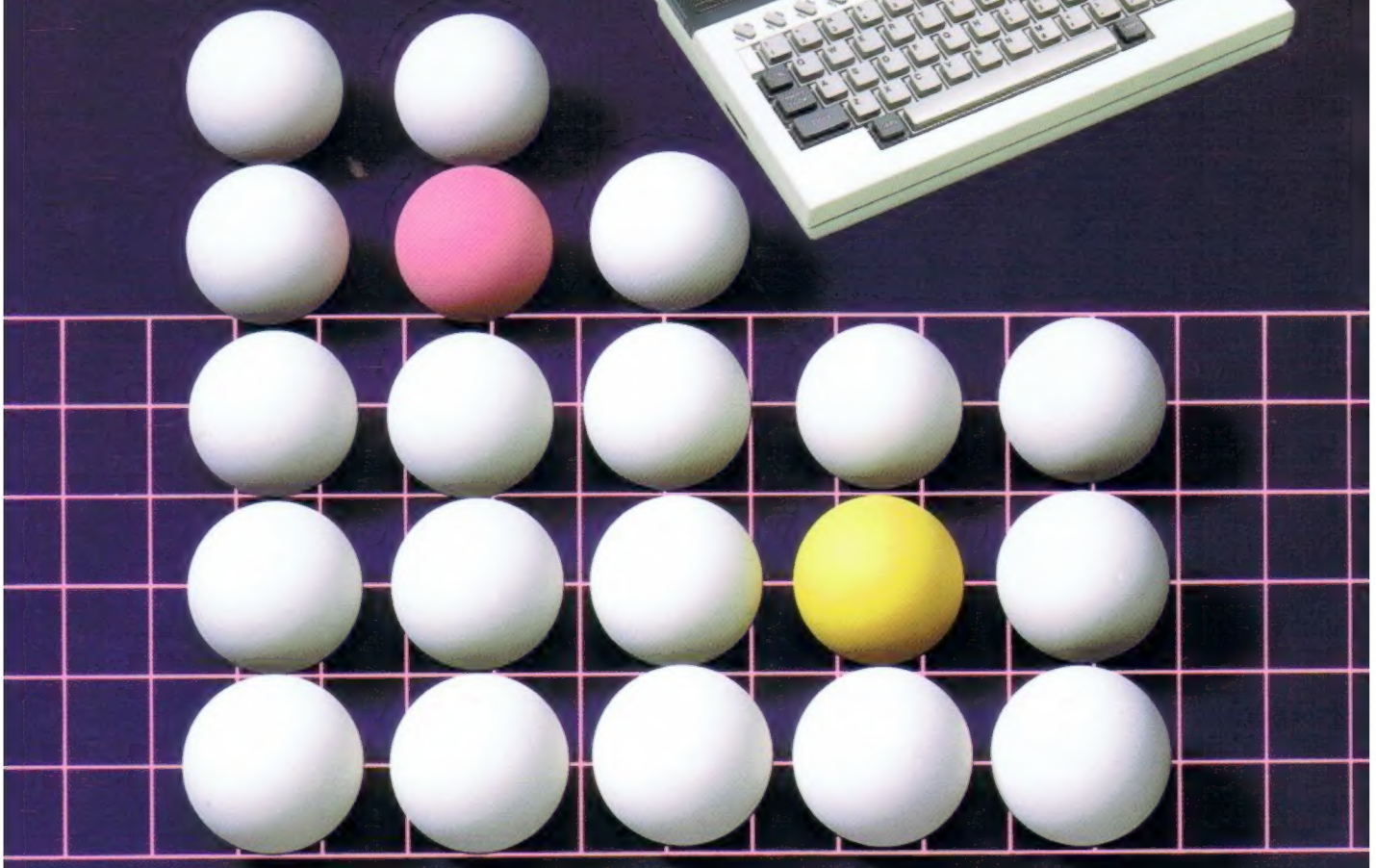
HANDY PERSONAL COMPUTER

CASIO FP-200

BEZUGSHANDBUCH

FÜR C_{85}-BASIC UND CETL



CASIO®

INHALT

1. **EINLEITUNG** ... *1*
1-1. Vorwort ... *1*
1-2. Eigenschaften von C_{85}-BASIC ... *2*
1-3. Tastenfeld ... *4*
1-3-1. Tastenfeldkonfiguration und Bedienung ... *4*
1-3-2. Betriebsartenwahltasten ... *4*
1-3-3. Programmierbare Funktionstasten ... *4*
1-3-4. Alphanumerische Tasten und Symboltasten ... *5*
1-4. Eingabebetriebsarten ... *6*
1-4-1. Logische Zeile und physikalische Zeile ... *8*
1-4-2. Tasteneingabe ... *8*
1-4-3. Direktbetriebsart ... *8*
1-3-5. Sonstige Tasten ... *9*

2. **VON C_{85}-BASIC GEHANDHABTE DATEN** ... *10*
2-1. Numerische Daten ... *10*
2-1-1. Reelle Zahlen einfacher Genauigkeit ... *10*
2-1-2. Reelle Zahlen doppelter Genauigkeit ... *10*
2-2. Zeichendaten ... *11*

3. **GRUNDLEGENDE INFORMATIONEN ÜBER C_{85}-BASIC** ... *12*
3-1. In Programmen verwendete Zeichen ... *12*
3-2. Reservierte Wörter ... *13*
3-3. Programme ... *14*
3-3-1. Programmbereiche ... *14*
3-3-2. Programmorganisation ... *14*
3-4. Zeilen ... *15*
3-4-1. Zeilennummer ... *15*
3-5. Sätze ... *16*
3-6. Konstanten ... *17*
3-6-1. Numerische Konstanten ... *17*
3-6-2. Zeichenkonstanten ... *18*
3-7. Variablen und Variablengruppen (Arrays) ... *19*
3-7-1. Variablenname und Variablengruppenname ... *19*
3-7-2. Variablengruppen ... *20*
3-7-3. Handhabung von Variablen und Variablengruppen ... *21*
3-8. Umwandlung von Datentypen ... *22*
3-9. Ausdrücke ... *23*
3-9-1. Numerische Ausdrücke ... *23*
3-9-1-1. Arithmetische Operation ... *23*
3-9-1-2. Vergleichsoperatoren ... *24*
3-9-1-3. Logische Operatoren ... *25*
3-9-1-4. Bewertung von numerischen Ausdrücken und Vorrang von Operationen ... *26*
3-9-2. Zeichenausdrücke ... *26*
3-10. Funktionen ... *27*
3-10-1. Durch den Benutzer definierte Funktionen (FN-Funktionen) ... *27*
3-10-2. Namen von durch den Benutzer definierten Funktionen ... *27*
3-11. In diesem Handbuch verwendete Notierung ... *28*

4. **BEFEHLE** 29
4-1. Systembefehle 29
4-1-1. PROG-Befehl 29
4-1-2. PASS-Befehl 30
4-1-3. NEW- und NEW-ALL-Befehl 31
4-1-4. SYSTEM-Befehl 32
4-1-5. KEY-Befehl 33
4-1-6. KEY-LIST-Befehl 35
4-1-7. CLEAR-Befehl 36
4-1-8. RESET-Befehl 38
4-1-9. AREA-Befehl 39
4-2. Programmredigieren 40
4-2-1. RENUM-Befehl 40
4-2-2. LIST-Befehl 42
4-2-3. EDIT-Befehl 44
4-3. Ausführungssteuerbefehle 46
4-3-1. RUN-Befehl 46
4-3-2. TRON-Befehl 47
4-3-3. TROFF-Befehl 48

5. **GRUNDLEGENDE OPERATIONEN UND FUNKTIONEN** 49
5-1. Ausführungssteuerung 49
5-1-1. END-Anweisung 49
5-1-2. STOP-Anweisung 50
5-1-3. GOTO-Anweisung 51
5-1-4. GOSUB-RETURN-Anweisung 52
5-1-4-1. GOSUB-Anweisung 52
5-1-4-2. RETURN-Anweisung 53
5-1-5. IF-THEN-ELSE-Anweisung 54
5-1-6. ON-GOTO-Anweisung 56
5-1-7. ON-GOSUB-Anweisung 58
5-1-8. FOR-NEXT-Anweisung 60
5-2. Kommentare 62
5-2-1. REM-Anweisung 62
5-3. Datenmanipulation 63
5-3-1. LET-Anweisung 63
5-4. Von einem Programm gelesene Daten 64
5-4-1. DATA-Anweisung 64
5-4-2. READ-Anweisung 65
5-4-3. RESTORE-Anweisung 67
5-5. Anzeige 69
5-5-1. PRINT-Anweisung 69
5-5-2. TAB-Funktion 73
5-5-3. PRINT-USING-Anweisung 75
5-5-4. LOCATE-Anweisung 78
5-5-5. CLS-Anweisung 79
5-6. Tastenfeldeingabe 80
5-6-1. INPUT-Anweisung 80
5-6-2. INKEYS-Funktion 82
5-7. Variablengruppen 83
5-7-1. OPTION-BASE-Anweisung 83

5-7-2. DIM-Anweisung ... 84
5-8. Maschinensprachenruf ... 86
5-8-1. CALL-Anweisung ... 86
5-9. Typerklärungen ... 87
5-9-1. DEF/SNG/DBL/STR-Anweisung ... 87
5-9-2. CSNG/CDBL-Funktionen ... 89
5-10. Mathematische Funktionen ... 90
5-10-1. Trigonometrische Funktionen ... 90
5-10-1-1. ANGLE-Anweisung ... 90
5-10-1-2. SIN/COS/TAN-Funktion ... 92
5-10-1-3. ASN/ACS/ATN-Funktionen ... 94
5-10-2. Logarithmische Funktionen und Exponentialfunktionen ... 96
5-10-2-1. LOG/LGT-Funktionen ... 96
5-10-2-2. EXP-Funktion ... 98
5-10-3. Sonstige arithmetische Funktionen ... 99
5-10-3-1. SQR-Funktion ... 99
5-10-3-2. ABS-Funktion ... 100
5-10-3-3. SGN-Funktion ... 101
5-10-3-4. INT-Funktion ... 102
5-10-3-5. FIX-Funktion ... 104
5-10-3-6. FRAC-Funktion ... 105
5-10-3-7. ROUND-Funktion ... 106
5-10-3-8. RND-Funktion ... 108
5-10-3-9. RANDOMIZE-Anweisung ... 109
5-11. Zeichendatenmanipulation ... 110
5-11-1. CHR$-Funktion ... 110
5-11-2. ASC-Funktion ... 111
5-11-3. STR$-Funktion ... 112
5-11-4. VAL-Funktion ... 113
5-11-5. MID$-Funktion ... 114
5-11-6. LEFT$-Funktion ... 115
5-11-7. RIGHT$-Funktion ... 116
5-11-8. LEN-Funktion ... 117
5-12. Funktionsdefinition ... 118
5-12-1. DEFFN-Anweisung ... 118
5-12-2. FN-Funktionen ... 119
5-13. Sonstige Funktionen ... 120
5-13-1. FRE-Funktion ... 120
5-13-2. PEEK-Funktion ... 121
5-13-3. POKE-Anweisung ... 122
5-13-4. DATE$- und TIME$-Funktion ... 123

6. STATISTISCHE VERARBEITUNG ... 124
6-1. STAT-CLEAR-Befehl ... 124
6-2. STAT-Befehl ... 125
6-3. Statistische Funktionen ... 126

7. GRAFIKEN ... 127
7-1. INIT-Anweisung ... 127
7-2. DRAW/DRAWC-Anweisungen ... 129
7-3. QUAD-Anweisung ... 130

7-4. POINT-Funktion ... 131

8. DRUCKERSTEUERUNG ... 132
8-1. LLIST-Anweisung ... 132
8-2. LPRINT-Anweisung ... 133
8-3. TAB-Funktion (LPRINT) ... 135
8-4. LPRINT-USING-Anweisung ... 136

9. DATEIVERARBEITUNG ... 137
9-1. Sequentielle Verarbeitung und Direktverarbeitung ... 137
9-1-1. MOUNT-Anweisung ... 138
9-1-2. OPEN-Anweisung ... 139
9-1-3. CLOSE-Anweisung ... 141
9-2. Sequentielle Verarbeitung ... 142
9-2-1. PRINT#-Anweisung ... 143
9-2-2. PRINT#USING-Anweisung ... 144
9-2-3. INPUT#-Anweisung ... 145
9-2-4. EOF-Funktion ... 146
9-3. Direktverarbeitung ... 148
9-3-1. FIELD-Befehl ... 150
9-3-2. GET-Anweisung ... 151
9-3-3. PUT-Anweisung ... 152
9-3-4. RSET-Anweisung ... 154
9-3-5. LSET-Anweisung ... 156
9-3-6. MKI$/MKS$/MKD$/MKF$-Funktionen ... 158
9-3-7. CVI/CVS/CVD/CVF-Funktionen ... 159
9-3-8. LOC-Funktion ... 161
9-3-9. LOF-Funktion ... 163
9-4. Dateihandhabung ... 165
9-4-1. FORMAT-Befehl ... 165
9-4-2. FILES-Befehl ... 166
9-4-3. KILL-Befehl ... 167
9-5. Programmdateien ... 168
9-5-1. SAVE-Befehl ... 169
9-5-2. LOAD-Befehl ... 170

10. KASSETTENTONBANDGERÄT (CAS 0:) ... 172
10-1. Dateideskriptor ... 172
10-2. Umriß der Verarbeitung ... 173
10-3. VERIFY-Befehl ... 175

11. FDD (FLOPPY DISK LAUFWERK) ... 176
11-1. Dateideskriptor ... 176
11-2. Umriß der Verarbeitung ... 177

12. DATENLEITUNGEN ... 179
12-1. Dateideskriptor ... 179
12-2. Umriß der Verarbeitung ... 180

13. CETL ... 181
13-1. Konzepte für CETL ... 181
13-1-1. Eigenschaften von CETL ... 181
13-1-2. Funktionen von CETL ... 181
13-2. Dateiorganisation ... 182
13-3. In CETL anwendbare Befehle ... 183
13-4. Eingabebereich für Datensatz und Posten ... 184
13-5. PF-Tasten in CETL-Betriebsart ... 185
13-6. CETL-Betriebsverfahren ... 186
13-7. CETL-Befehle ... 187
13-7-1. N (neue Datei) ... 187
13-7-2. A (automatische Eingabe) ... 188
13-7-3. I (Datei einschieben) ... 189
13-7-4. D (Datei löschen) ... 190
13-7-5. M (Datei bewegen) ... 191
13-7-6. R (Neubenennung) ... 192
13-7-7. K (Abbruch) ... 193
13-7-8. B (Leerstelle) ... 194
13-7-9. S (Sortieren) ... 196
13-7-10. F (Finden) ... 198
13-7-11. J (Sprung) ... 200
13-7-12. L (Liste) ... 201
13-7-13. T (Tabelle) ... 202
13-7-14. C (Berechnen) ... 204
13-7-15. P (Schreiben) ... 205
13-7-16. G (Lesen) ... 206
13-8. CETL-Verwaltungsfunktionen ... 207
13-8-1. RC-Funktion ... 208
13-8-2. IT-Funktion ... 209
13-8-3. RC ( )-Funktion ... 210
13-8-4. IT ( )-Funktion ... 211
13-8-5. FL-Funktion ... 212
13-8-6. SUMRC-Funktion ... 214
13-8-7. SUMIT-Funktion ... 216
13-9. CETL und BASIC ... 218
13-10. Gemeinsame Benutzung von sequentiellen Dateien ... 219

TASTENFELDEINGABETABELLE ... 220
ERKLÄRUNG DER TASTENFUNKTIONEN ... 225
TASTATUR-LAYOUT ... 227
BEFEHLE, ANWEISUNGEN UND FUNKTIONEN FÜR C_{85}-BASIC ... 236
CETL-FUNKTIONEN ... 249
FEHLERKODETABELLE ... 251
INDEX DER BEFEHLE, ANWEISUNGEN UND FUNKTIONEN ... 253
C_{85}-BASIC SPEZIFIKATION ... 258

1. EINLEITUNG

1-1. Vorwort

Dieses Handbuch beschreibt die Programmiersprache C_{85}-BASIC und CETL für den FP-200. Zusätzlich zu diesem Handbuch sind die folgenden Handbücher für den FP-200 herausgegeben worden.

(1) Bedienungsanleitung

Dieses Handbuch bietet grundlegende Informationen für den FP-200. Lesen Sie unbedingt zuerst dieses Handbuch vollständig durch.

1-2. Eigenschaften von C_{85}-BASIC

C_{85}-BASIC ist eine leistungsfähige Version der Standardsprache BASIC mit erweiterten Rechen- und Dateiverarbeitungsfunktionen.

Wie die Standardsprache BASIC hat auch C_{85}-BASIC die folgenden Eigenschaften:

(1) Leicht zu lernende Sprache

Mit weniger komplizierter Satzlehre ist C_{85}-BASIC leichter zu lernen als FORTRAN und andere Programmiersprachen.

(2) Einfache Programmierung

Programme können durch Zusammenarbeit mit dem Computer wirksam und leicht vorbereitet, modifiziert und ausgeführt werden.

(3) Arithmetische Operationen mit hoher Genauigkeit

Es sind vier numerische Typen erhältlich, um die gewünschte Operationsgenauigkeit zu erhalten.

1) Reeller Typ einfacher Genauigkeit
   9 Stellen werden intern verwendet, von denen 6 angezeigt werden können.
2) Reeller Typ doppelter Genauigkeit
   19 Stellen werden intern verwendet, von denen 16 angezeigt werden können.

Da der Exponententeil jedes reellen Typs im Bereich von −99 bis 99 liegen kann, können praktisch jegliche numerischen Daten gehandhabt werden.

(4) Weitgehende mathematische und statistische Funktionen

1) 17 typische mathematische Funktionen können in einfacher oder doppelter Genauigkeit verwendet werden:

   SIN, COS, TAN, ASN, ACN, ATN, LOG, LGT, EXP, SQR, ABS, SGN, INT, FIX, FRAC, ROUND, RND*.

   Die Funktion RND, kann nur in einfacher Genauigkeit verwendet werden.

2) Leistungsfähige Zeichenmanipulationsfunktionen:

CHR$, STR$, MID$, LEFT$, RIGHT$, ASC, VAL, LEN.

3) Statistische Funktionen doppelter Genauigkeit, einschließlich linearer Regression.

(5) Arithmetische Funktion mit binärkodierten Dezimalzahlen

Diese Funktion kann zuverlässig für geschäftliche und wissenschaftliche Anwendungen verwendet werden.

(6) Unterteilung des Programmbereichs

Bis zu 10 verschiedene Programme konnen gleichzeitig im Hauptspeicher für anschließende Ausführung gespeichert werden, wodurch die Notwendigkeit beseitigt wird, Programme für jeden Lauf zu laden.

(7) Erweiterte Variablennamen

Variablennamen konnen eine Lange von bis zu 255 Zeichen haben. Sie konnen Kleinbuchstaben enthalten. Verwendung von Variablennamen, die die Bedeutung der entsprechenden Variablen klar repräsentieren, macht das Programm leicht verständlich.

(8) Leistungsfähige Fehlersuchfunktion

Es sind zwei Befehle zugefügt worden, um den Weg der Programmausführung verfolgen zu können: durch den TRON-Befehl wird die Nummer der Zeile, die ausgeführt wird, auf der Flüssigkristallanzeige angezeigt.

(9) Standardisierte Handhabung externer Geräte

Im Datenaustausch mit den verschiedenen externen Geraten (FDD usw.) wurd das Konzept der Datei-Handhabung eingeführt. Da der Datenaustausch mit externen Geräten uber eine Datei durchgeführt wird, können alle Geräte fast in der gleichen Weise gehandhabt werden.

(10) Grafiken

1) Benutzerkoordinatensystem

Dieses System ermoglicht es dem Benutzer, jede beliebige Position auf dem Bildschirm in durch den Benutzer definierten Koordinaten zu bezeichnen. Dies beseitigt die Notwendigkeit komplizierter Koordinatenberechnungen und ermoglicht einfachere Grafikverarbeitung.

2) Leistungsfähige Grafikanweisungen

Diese Anweisungen können bequem verwendet werden, um Geraden, Rechtecke, Kreise usw. zu zeichnen.

1-3. Tastenfeld

1-3-1. Tastenfeldkonfiguration und Bedienung

Der FP-200 hat ein mit ASCII vertragliches, leicht zu verwendendes Tastenfeld, das Eingabe einer Serie von Tastenbetätigungen durch Druck auf eine Taste ermöglicht, unabhängige Tasten für häufig verwendete Funktionen hat usw.

1-3-2. Betriebsartenwahltasten

CTRL	Steuertaste	Betätigung einer Taste bei gedrückter Steuertaste verursacht Eingabe in Steuerbetriebsart.
GRAPH	Grafiktaste	Betätigung einer Taste bei gedrückter Grafiktaste verursacht Eingabe in Grafikbetriebsart.
CAPS	Großbuchstabenumschalttaste	Durch diese Taste wird die Großbuchstaben-Umschaltbetriebsart ein- bzw. ausgeschaltet. Bei jeder Betätigung dieser Taste wird die Betriebsart umgeschaltet (Ein bzw. Aus). Wenn diese Taste eingeschaltet ist (Großbuchstaben-Umschaltbetriebsart), so wechselt der Positionsanzeiger von "–" zu "=". Diese Taste wird zur Eingabe von Buchstaben verwendet.
SHIFT	Umschalttaste	Betätigung einer Taste bei gedrückter Umschalttaste verursacht Hochschaltungseingabe in Niederschaltungsbetriebsart. Beachten Sie bitte, daß diese Taste in Steuerbetriebsart und in Grafikbetriebsart unwirksam ist.
SHIFT LOCK	Umschaltverriegelungstaste	Diese Taste wird zum Verriegeln und Entriegeln der Umschaltbetriebsart verwendet. Bei jeder Betätigung dieser Taste wird die Umschaltbetriebsart verriegelt bzw. entriegelt. Wenn die Umschaltbetriebsart verriegelt ist, so ändert sich der Positionsanzeiger von "–" zu "=" oder von "=" zu "=".

1-3-3. Programmierbare Funktionstasten

Die in der obersten Reihe des Tastenfelds angeordneten Tasten PF0 bis PF9 werden als programmierbare Funktionstasten bezeichnet. Diese Tasten ermöglichen Ausführung einer Anzahl von im voraus programmierten Tastenbetätigungen durch einen einzelnen Tastendruck. Die Anfangseinstellung dieser Tasten ist wie folgt:

BASIC-Betriebsart

PF 0 : EDIT	PF 4 : RUN$^{C}_{R}$	PF 7 : LOAD"
PF 1 : PROG	PF 5 : S_0	PF 8 : SAVE"
PF 2 : SYSTEM$^{C}_{R}$	PF 6 : FILES$^{C}_{R}$	PF 9 : P. DATE$, TIME$
PF 3 : LIST$^{C}_{R}$		

CETL-Betriebsart

PF 0 : D_3	PF 4 : P. FRE$^{C}_{R}$	PF 7 : RC (
PF 1 : FILE	PF 5 : S_0	PF 8 : IT (
PF 2 : SYSTEM$^{C}_{R}$	PF 6 : FL (	PF 9 : P. DATE$, TIME$
PF 3 : S_H		

* S_0: CTRL + N (Zehnertasten-Betriebsart)
  D_3: CTRL + S (Edit-Betriebsart)
  S_H: CTRL + A (Kommando-Menü)

(Die Tasten PF5 bis PF9 sind die Tasten PF0 bis PF4 bei gedrückter Umschalttaste.)

* PF5 (d.h. **SHIFT + FP0**) ermöglicht die Verwendung der alternativen numerischen Tasten für die Zahleneingabe. Desgleichen können die : und ; Tasten zur Eingabe von + und * verwendet werden.

1-3-4. Alphanumerische Tasten und Symboltasten

Diese Tasten werden verwendet, um alphanumerische Zeichen, Symbole usw. einzugeben. In Steuerbetriebsart bewirken sie vorherbestimmte Operationen. Jede Taste hat ein oder zwei Zeichen, von denen jedes durch Betätigung der Taste in der entsprechenden Betriebsart eingegeben werden kann. Um das einfache Aussehen des Tastenfelds zu erhalten, sind Grafiksymbole und Operationen in Steuerbetriebsart nicht auf den Tasten gezeigt.

(1) Betrieb in normaler Betriebsart

1) Wenn die Umschalttaste nicht gedrückt ist:

Die Tasten A bis Z geben die entsprechenden Großbuchstaben ein. Für andere Tasten wird das entsprechende niedriggestellte Zeichen eingegeben.

Beispiel

2) Wenn die Umschalttaste gedrückt ist:

Die Tasten A bis Z geben die entsprechenden Kleinbuchstaben ein. Für andere Tasten wird das entsprechende hochgestellte Zeichen eingegeben.

Beispiel

(2) Betrieb in Großbuchstaben-Umschaltbetriebsart

1) Wenn die Umschalttaste nicht gedrückt ist:

Die Tasten A bis Z geben die entsprechenden Kleinbuchstaben ein. Für andere Tasten wird das entsprechende niedriggestellte Zeichen eingegeben.

Beispiel

2) Wenn die Umschalttaste gedrückt ist:

Die Tasten A bis Z geben die entsprechenden Großbuchstaben ein. Für andere Tasten wird das entsprechende hochgestellte Zeichen eingegeben.

Beispiel

(3) Betrieb in Grafikbetriebsart

Ein vorherbestimmtes Grafiksymbol wird eingegeben, unabhängig davon ob die Umschalttaste gedrückt ist oder nicht.

Beispiel:

(4) Betrieb in Steuerbetriebsart

Ein vorherbestimmter Betrieb wird durchgeführt. Beziehen Sie sich auf die Tastenkodetabelle am Ende dieses Handbuchs.

(5) Betrieb in Umschaltverriegelungsbetriebsart

Die Umschaltbetriebsart ist verriegelt, unabhängig davon ob die Umschalttaste gedrückt ist oder nicht.

Beispiel:

(6) Zifferntastenfeld

Durch Druck auf PF 5 können die folgenden Tasten als Zifferntasten bzw. Rechensymbole verwendet werden.

U→4 I→5 O→6

J→1 K→2 L→3 ;→+ :→*

M→0

1-3-5. Sonstige Tasten

Abbruchtaste — Diese Taste verursacht Beendigung der Programmausführung.

Unterbrechungstaste — Durch Druck auf diese Taste während des Betriebs wird der Betrieb zeitweilig unterbrochen. Durch erneuten Druck auf diese Taste wird der Betrieb wieder aufgenommen.

Rücklauftaste — Drücken Sie diese Taste, nachdem Sie dem Computer einen Befehl gegeben haben oder nach Eingabe eines Programms oder Daten. Der Inhalt der gegenwärtig durch den Positionsanzeiger angezeigten Zeile (logische Zeile) wird dann eingegeben.

Taste für Aufwärtsbewegung des Positionsanzeigers — Hierdurch wird der Positionsanzeiger um eine Zeile nach oben bewegt. Durch Druck auf diese Taste in BASIC-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der vorhergehenden Zeile über der gegenwärtigen Zeile in Redigierbetriebsart eingeleitet. Durch Druck auf diese Taste in CETL-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der vorhergehenden Zelle über der gegenwärtigen Zelle in Redigierbetriebsart eingeleitet.

Taste für Abwärtsbewegung des Positionsanzeigers

Durch Druck auf diese Taste wird der Positionsanzeiger um eine Zeile nach unten bewegt. Durch Druck auf diese Taste in BASIC-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der auf die gegenwärtige Zeile folgenden Zeile in Redigierbetriebsart eingeleitet. Durch Druck auf diese Taste in CETL-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der auf die gegenwärtige Zelle folgenden Zelle in Redigierbetriebsart eingeleitet.

Taste für Linksbewegung des Positionsanzeigers

Durch Druck auf diese Taste wird der Positionsanzeiger um eine Schreibstelle nach links bewegt. Durch Druck auf diese Taste in BASIC-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der gegenwärtigen Zeile in Redigierbetriebsart eingeleitet. Durch Druck auf diese Taste in CETL-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der Zelle links von der gegenwärtigen Zelle in Redigierbetriebsart eingeleitet.

Taste für Rechtsbewegung des Positionsanzeigers

Durch Druck auf diese Taste wird der Positionsanzeiger um eine Schreibstelle nach rechts bewegt. Durch Druck auf diese Taste in BASIC-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der auf die gegenwärtige Zeile folgenden Zeile in Redigierbetriebsart eingeleitet. Durch Druck auf diese Taste in CETL-Betriebsart bei gedrückter Taste SHIFT wird Redigieren der Zelle rechts von der gegenwärtigen Zelle in Redigierbetriebsart eingeleitet.

Einschubtaste/ Löschtaste

Durch Druck auf diese Taste wird das Zeichen an der gegenwärtigen Position des Positionsanzeigers gelöscht, und alle Zeichen nach der gegenwärtigen Position des Positionsanzeigers werden um eine Schreibstelle nach links bewegt. Durch Druck auf diese Taste bei gedrückter Umschalttaste werden alle Zeichen nach der gegenwärtigen Position des Positionsanzeigers um eine Schreibstelle nach rechts bewegt.

Löschen des Bildschirms/Rückkehr des Positionsanzeigers zur Ausgangsposition

Durch Druck auf diese Taste wird der Positionsanzeiger zur linken oberen Ecke des Bildschirms bewegt. Durch Druck auf diese Taste bei gedrückter Umschalttaste wird die gesamte Bildschirmanzeige gelöscht, und der Positionsanzeiger wird zur linken oberen Ecke des Bildschirms bewegt. Wenn die Taste für Löschen des Bildschirms bzw. Rückkehr des Positionsanzeigers zur Ausgangsposition in Redigierbetriebsart gedrückt wird, so werden alle in der gerade redigierten Zeile gemachten Modifikationen ungültig gemacht.

1-4. Eingabebetriebsarten

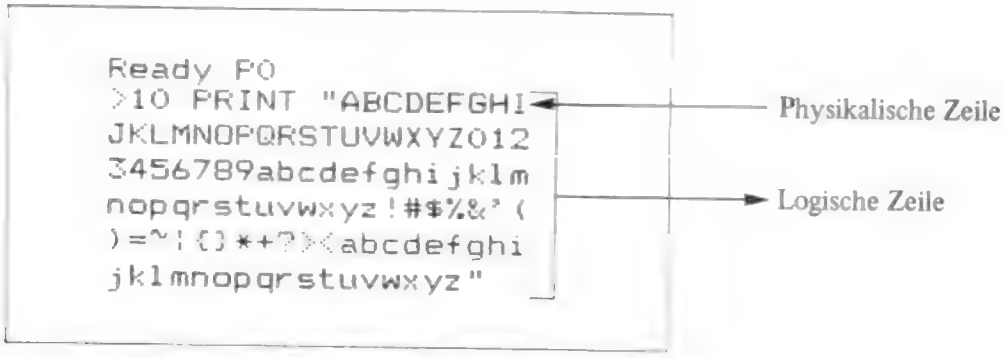
1-4-1. Logische Zeile und physikalische Zeile

Wenn Tasten gedruckt werden, so wird der Inhalt dieser Tasten auf dem Bildschirm angezeigt. Dieser Inhalt kann durch Verwendung der Positionsanzeigertasten, der Einschubtaste und der Loschtaste modifiziert werden. Dann kann der Inhalt durch Druck auf die Rücklauftaste oder die Eingabetaste zum Computer eingegeben werden.

Beachten Sie jedoch, daß der Umfang der Daten, die durch die Rucklauftaste eingegeben werden können (der gesamte Bildschirm voll Daten oder ein einziges Zeichen an der gegenwärtigen Position des Positionsanzeigers) noch festgelegt werden muß.

Zu diesem Zweck wird der Umfang der einzugebenden Daten als die Zeile definiert, in der sich der Positionsanzeiger befindet. Wenn diese Zeile eine Zeile auf dem Bildschirm ist, so können jeweils nur 20 Zeichen eingegeben werden.

Jede Zeile auf dem Bildschirm wird als physikalische Zeile bezeichnet. Die tatsächlich zur Eingabe verwendete Zeile hat jedoch einen weiteren Umfang und wird als logische Zeile bezeichnet. Die logische Zeile schließt alle eingegebenen Daten bis zum Drucken der Rücklauftaste ein.



Diese Daten schließen auch Leerstellen ein. Durch Druck auf die Leertaste kann eine Leerstelle in die logische Zeile eingeschlossen werden.

1-4-2. Tasteneingabe

1-4-2-1. Gültige Position des Positionsanzeigers für Tastenfeldeingabe

Wenn Daten vom Tastenfeld her eingegeben werden, so können Zeichen (Zeichenkode 20 bis 7E und 80 bis FE) nicht eingegeben werden, wenn die Position des Positionsanzeigers über der logischen Zeile ist. Wenn das Ende der logischen Zeile auf dem Bildschirm ist, so kann die Taste ↓ nicht in der untersten Zeile und die Taste → nicht am rechten Ende der Zeile verwendet werden.

1-4-2-2. Zeilenvorschub und Aufrollen der Eingabezeile

Wenn der Anfang einer Zeile an der Schreibstelle 0 angenommen wird, wo tritt ein Zeilenvorschub an der Schreibstelle 19 ein. Wenn ein Zeichen an der Schreibstelle 19 auf der untersten Zeile des Bildschirms eingegeben wird, so wird der Bildschirm um eine Zeile nach oben gerollt. Während Eingabe vom Tastenfeld findet bei Verwendung der Einschubtaste kein Aufrollen statt, selbst wenn die Daten auf dem Bildschirm die Bildschirmkapazität (180 Zeichen) überschreiten. Aufrollen erfolgt nur bei Eingabe eines Zeichens an der rechten unteren Ecke des Bildschirms.

1-4-2-3. Betrieb der Taste für Löschen des Bildschirms/Rückkehr des Positionsanzeigers zur Ausgangsposition während Tastenfeldeingabe

Während Tastenfeldeingabe werden die schon eingegebenen Zeichen gelöscht und nicht beachtet, wenn die Taste für Löschen des Bildschirms/Rückkehr des Positionsanzeigers zur Ausgangsposition gedrückt wird. Die Daten verbleiben jedoch auf dem Bildschirm.

1-4-3. Direktbetriebsart

Durch Druck der Rücklauftaste (RETURN) nach Eingabe einer Anweisung wird die Anweisung sofort ausgeführt. Auf diese Weise können einfache Operationen durchgeführt werden, ohne daß Programmieren erforderlich ist.

Beispiel

PRINT "Future Personal Computer"
RETURN

```
>PRINT "Future Perso
nal Computer"
Future Personal Comp
uter
```

2. VON C_{85}-BASIC GEHANDHABTE DATEN

2-1. Numerische Daten

Die von C_{85}-BASIC gehandhabten numerischen Daten schließen ganze Zahlen, reelle Zahlen mit einfacher Genauigkeit, reelle Zahlen mit doppelter Genauigkeit und reelle Zahlen mit erweiterter Genauigkeit ein.

2-1-1. Reelle Zahlen einfacher Genauigkeit

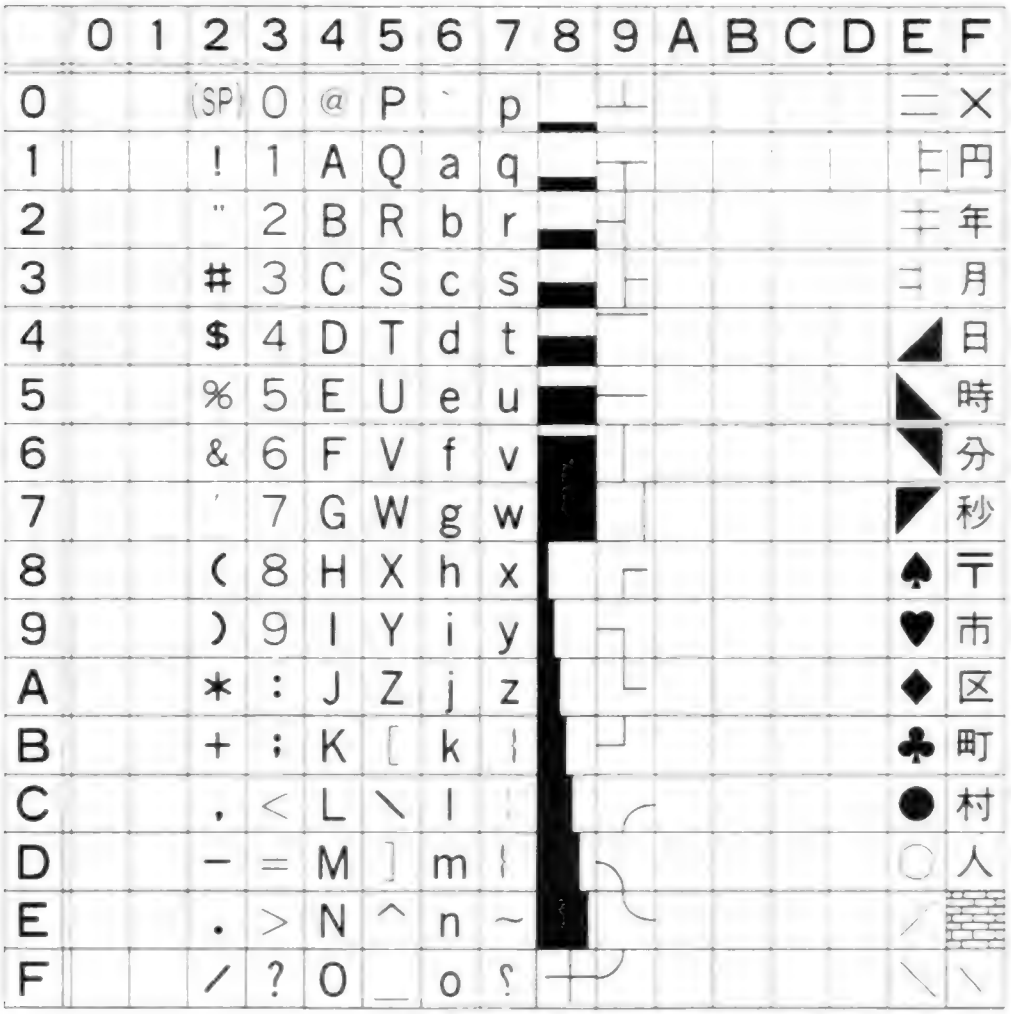
Bereich:	$\pm(1 \times 10^{-99}$ bis $9{,}99999999 \times 10^{99})$ und 0.
Mantisse:	9 Stellen
Exponent:	2 Stellen (ganze Zahlen von –99 bis 99)
Speicheranforderung:	6 Bytes

2-1-2. Reelle Zehlen doppelter Genauigkeit

Bereich:	$\pm(1 \times 10^{-99}$ bis $9{,}999999999999999999 \times 10^{99})$ und 0.
Mantisse:	19 Stellen
Exponent:	2 Stellen (ganze Zahlen von –99 bis 99)
Speicheranforderung:	11 Bytes

2-2. Zeichendaten

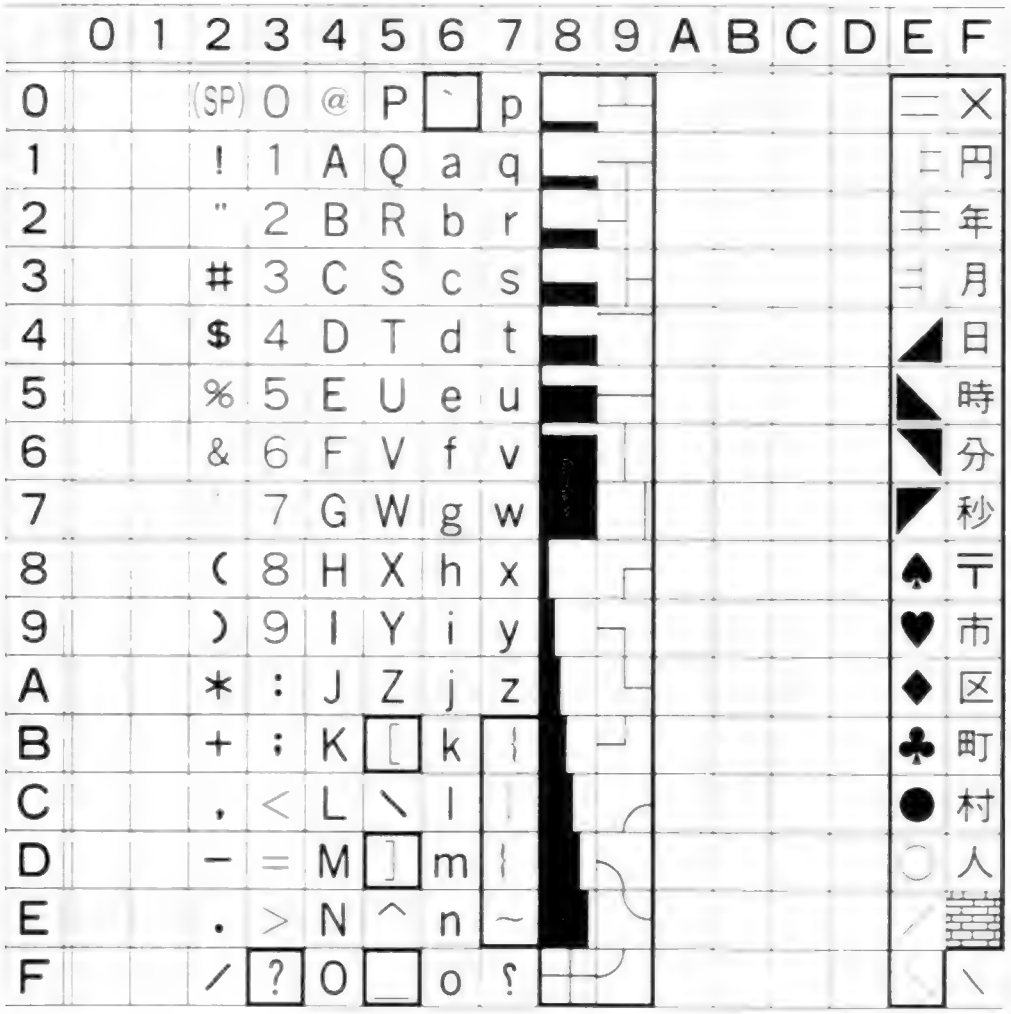
C_{85}-BASIC kann Zeichenketten mit variabler Länge von 0 bis 255 Zeichen handhaben. Ein Zeichen wird durch ein Byte ausgedrückt, und die meisten Zeichen entsprechen dem ASCII 8-Bit-Kode. Zeichendaten werden in dem durch die CLEAR-Anweisung festgelegten Zeichenbereich gespeichert.



3. GRUNDLEGENDE INFORMATIONEN ÜBER C_{85}-BASIC

3-1. In Programmen verwendete Zeichen

C_{85}-BASIC verwendet die folgenden Zeichen (20-7F, 80-9F und E0-FF im ASCII-Kode). Die dick umrandeten Zeichen (3F, 5B, 5D, 5F, 60, 7B-7E, 80-9F und E0-FE im ASCII-Kode) können nur in der REM-Anweisung, in der DATA-Anweisung und als Zeichenkonstanten verwendet werden.



3-2. Reservierte Wörter

Die nachfolgend aufgeführten Wörter werden reservierte Wörter genannt, und sie werden für besondere Zwecke (z.B. zur Anzeige von Befehlen, Funktionen usw.) verwendet. Für reservierte Wörter können Groß- oder Kleinbuchstaben verwendet werden, aber Kleinbuchstaben werden in die entsprechenden Großbuchstaben umgewandelt.

Beispiel

PRINT und print sind das gleiche reservierte Wort.

ABS	ACS	ALL	AND	ANGLE
AREA	AS	ASC	ASN	ATN
BASE	CALL	CDBL	CHR$	CLEAR
CLOSE	CLS	CNT	COS	CSNG
CVD	CVS	DATA	DTAE$	DEF
DEFDBL	DEFSNG	DEFSTR	DIM	DRAW
EDIT	ELSE	END	EOF	EXP
FIELD	FILE	FILES	FIX	FL
FN	FOR	FORMAT	FRAC	FRE
GET	GOSUB	GOTO	IF	INIT
INKEY$	INPUT	INT	IT	KEY
KILL	LEFT$	LEN	LET	LGT
LIST	LLIST	LOAD	LOC	LOCATE
LOF	LOG	LPRINT	LRA	LRB
LSET	MEANX	MEANY	MID$	MKD$
MKS$	MOD	MOUNT	NEW	NEXT
NOT	ON	OPEN	OPTION	OR
OUTPUT	PASS	PEEK	POINT	POKE
PRINT	PROG	PUT	QUAD	RANDOMIZE
RC	READ	REM	RENUM	RESET
RESTORE	RETURN	RIGHT$	RND	ROUND
RSET	RUN	SAVE	SDX	SDXN
SDY	SDYN	SGN	SIN	SQR
STAT	STEP	STOP	STR$	SUMIT
SUMRC	SUMX	SUMX2	SUMXY	SUMY
SUMY2	SYSTEM	TAB	TAN	THEN
TIME$	TO	TROFF	TRON	USING
VAL	VERIFY	XOR		

3-3. Programme

3-3-1. Programmbereiche

Bis zu 10 Programmbereiche, PROG 0 bis PROG 9, können zum Speichern von 10 verschiedenen Programmen verwendet werden. Die Länge jedes Programmbereichs wird automatisch entsprechend der Programmlänge bestimmt. Jedes Programm kann unabhängige Zeilennummern haben, aber die Variablen sind gemeinsam für alle Programmbereiche.

Durch den PROG-Befehl können die Programmbereiche manuell umgeschaltet werden. Durch die GOTO-Anweisung, die GOSUB-Anweisung usw. kann die Steuerung während der Programmausführung von einem Programmbereich zu einem anderen Programmbereich übertragen werden.

3-3-2. Programmorganisation

Ein Programm ist eine Ansammlung von Zeilen. Normalerweise werden Programmzeilen der Reihe nach in der Reihenfolge der Zeilennummern ausgeführt.

3-4. Zeilen

Eine Zeile besteht aus einer Zeilennummer und einer Anweisung (bzw. mehreren Anweisungen. Wenn mehr als eine Anweisung verwendet wird, so trennen Sie die Anweisungen durch einen Doppelpunkt (:). Die Zeile erscheint deshalb wie folgt:

Zeilennummer **Anweisung (: Anweisung)***

Die Klammern zeigen an, daß ihr Inhalt optional ist. Der Stern (*) zeigt an, daß das Element direkt vor dem Stern beliebig oft wiederholt werden kann. Beziehen Sie sich für weitere Einzelheiten auf den Abschnitt 3-11.

Beispiel 10 PRINT : PRINT A

Die Zeilenlange ist maximal 255 Zeichen, die durch einen LIST-Befehl angezeigt werden konnen.

3-4-1. Zeilennummer

Jede Zeile muß mit einer Zeilennummer beginnen. In der GOTO-Anweisung und in anderen Anweisungen wird auf die Zeilennummer Bezug genommen. Eine Zeilennummer muß eine ganze Zahl im Bereich von 1 bis 64999 sein.

In der Zeilennummer enthaltene Leerstellen werden nicht beachtet.

Beispiel

1 0 PRINT : GOTO 1 0 wird als 10 PRINT : GOTO 10 angesehen.
10 PRINT : GOTO 10

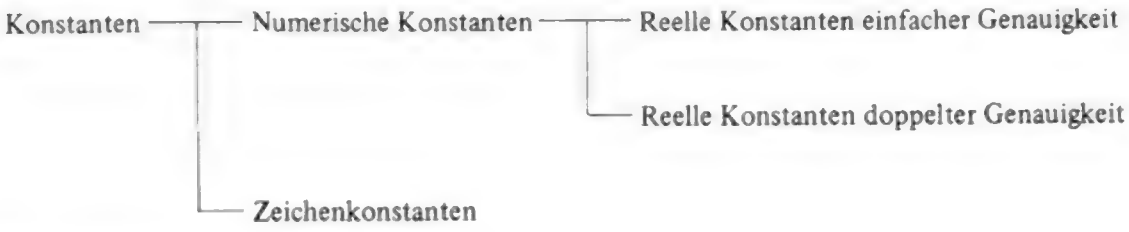
3-5. Sätze

Ein Satz beginnt mit einer Anweisung oder einem Befehl zur Anzeige der Art des Satzes. Die Satzlehre für einen Satz hängt vom Typ des Satzes ab.

Für C_{85}-BASIC werden die Begriffe "Befehl" und "Anweisung" oft austauschbar verwendet, obwohl der Begriff "Befehl" normalerweise in Direktbetriebsart und der Begriff "Anweisung" in Programmen bevorzugt wird.

3-6. Konstanten

Nummern oder Ketten, die literal in einem Programm erscheinen, werden als Konstanten bezeichnet.



3-6-1. Numerische Konstanten

Numerische Konstanten werden in einer der folgenden drei Notierungen angezeigt. Die Vorzeichen Plus (+) und Minus (−) konnen mit jeder Notierung verwendet werden. Eine beliebige Anzahl dieser Zeichen kann aufeinanderfolgend verwendet werden.

Wenn mehr als ein Vorzeichen verwendet wird, so wird nur die Anzahl der Minuszeichen gezählt, und eine ungerade Anzahl von Minuszeichen wird durch ein einziges Minuszeichen ersetzt. Wenn die Anzahl der Minuszeichen 0 oder eine gerade Zahl ist, so werden diese Minuszeichen durch ein einziges Pluszeichen ersetzt (die grundlegende mathematische Regel gilt). Leerstellen zwischen Vorzeichen oder zwischen einem Vorzeichen und einer Zahl werden nicht beachtet.

(1) Ganzzahlige Notierung

Eine oder mehrere Stellen nach einer beliebigen Anzahl von Plus- und/oder Minuszeichen.

Beispiel

```
  125
 −52
−−+−45      (=−45)
```

(2) Festkommanotierung

Eine Anzahl von Plus- und/oder Minuszeichen, gefolgt von einem ganzzahligen Teil, einem Dezimalpunkt (Komma) und einem Bruchteil in dieser Reihenfolge. Der ganzzahlige Teil und der Bruchteil können eine beliebige Anzahl von Stellen haben.

Beispiel

```
 +1.23
+−  .45      (=−0.45)
 2.37
 2.          (=2)
```

(3) Gleitkommanotierung

Eine Festkommazahl gefolgt von einem Exponenten. Der Exponent besteht aus einem E, einer beliebigen Anzahl von Plus- und/oder Minuszeichen und einer ein-oder zweistelligen Zahl. Dieser Teil zeigt eine Potenz von 10 an. Der Wert einer auf diese Weise angegebenen Zahl ist deshalb der Wert der durch Festkommanotierung gezeigten Zahl x 10 hoch Exponent.

Beispiel 1E−1 $(1\times10^{-1}=0.1)$

2. 3E +3 $(2.3\times10^{+3}=2300)$

Jede der durch eine der oben beschriebenen Notierungen dargestellten numerischen Konstanten wird entsprechend den nachfolgend beschriebenen Regeln als reelle Konstante einfacher Genauigkeit behandelt.

(1) Wenn die numerische Konstante ein nachgestelltes Symbol zur Anzeige des Typs hat, so wird sie als eine numerische Konstante dieses Typs behandelt.

!	Reeller Typ einfacher Genauigkeit
#	Reeller Typ doppelter Genauigkeit

Wenn die Anzahl der Stellen einer numerischen Konstante die festgelegte Anzahl der Stellen der Genauigkeit überschreitet, so werden die überzähligen Stellen vernachlässigt.

Beispiel		
	9.9999999832E3!	Reelle Konstante einfacher Genauigkeit von $9{,}99999998 \times 10^3$
	23.5#	Reelle Konstante doppelter Genauigkeit von 23,5

(2) In dezimaler (oder ganzzahliger) Notierung dargestellte numerische Konstanten ohne Typbezeichnung werden wie folgt behandelt:

- Als reelle Konstanten einfacher Genauigkeit, wenn die Stellenzahl 9 oder weniger ist.
- Als reelle Konstanten doppelter Genauigkeit, wenn die Stellenzahl 10 bis 19 ist.

Beachten Sie jedoch bitte, daß vorangehende Nullen (z.B. 00 bei 0012) nicht als Stellen gezählt werden.

Beispiel		
	12.57	Konstante einfacher Genauigkeit
	17900000025	Konstante doppelter Genauigkeit
	45.2000000000	Konstante doppelter Genauigkeit
	1234567801 !	Konstante einfacher Genauigkeit

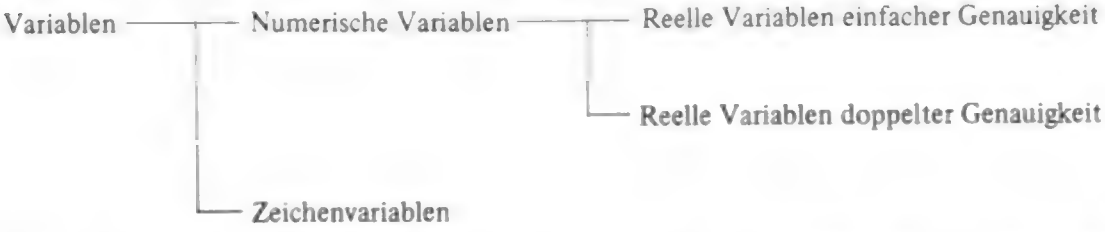
3-6-2. Zeichenkonstanten

Eine Zeichenkonstante ist eine Folge von in Anführungszeichen (") eingeschlossenen Zeichen. In den meisten Fällen kann das rechte Anführungszeichen nach dem letzten Zeichen in einer logischen Zeile ausgelassen werden. In diesem Fall wird das letzte Zeichen, ausgenommen Leerstellen, als das Ende der Konstanten angenommen. Nicht in Programmen verwendete Zeichen (interne Kode 00-1F, 7F, A0-DF, FF) und das Anführungszeichen (") können nicht als Zeichenkonstanten eingeschlossen werden.

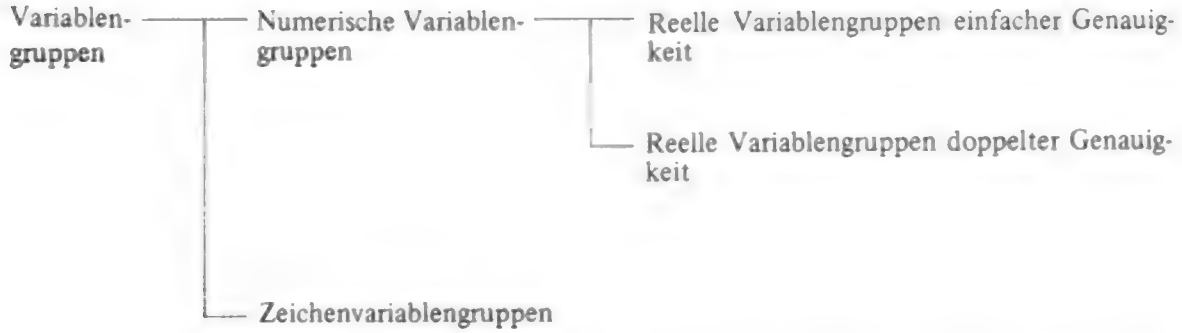
Beispiel		
	"ABCDEF"	
	"CASIOABC	(das rechte Anführungszeichen wird ausgelassen)
	""	(Anzeige einer Zeichenkette der Länge 0)

3-7. Variablen und Variablengruppen

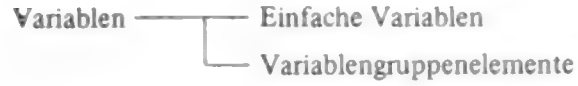
Variablen bieten eine Moglichkeit, auf numerische Daten oder Zeichendaten mit einem Namen Bezug zu nehmen. Die Variablen sind wie folgt klassifiziert:



Eine Variablengruppe ist eine Ansammlung von Variablen des gleichen Typs, auf die kollektiv unter einem gemeinsamen Namen Bezug genommen wird. Die Variablengruppen sind wie folgt klassifiziert:



Die individuellen Variablen einer Variablengruppe werden als Variablengruppenelemente bezeichnet, während Variablen, die nicht Teil einer Variablengruppe sind, als einfache Variablen bezeichnet werden. Variablen können deshalb wie folgt klassifiziert werden.



3-7-1. Variablenname und Variablengruppenname

Variablennamen und Variablengruppennamen folgen den gleichen, nachfolgend beschriebenen Regeln.

(1) Sie müssen eine Zeichenkette sein, die mit einem Großbuchstaben (interner Kode 41-5A) oder einem Kleinbuchstaben (61-7A) beginnt.

(2) Sie müssen aus Großbuchstaben, Kleinbuchstaben oder Zahlen (interner Kode 30-39) bestehen.

(3) Sie dürfen nicht mit einem reservierten Wort beginnen.

(4) Die Länge darf 255 Zeichen nicht überschreiten.

Beispiel	ABC	o	
	prIME	o	
	FORSIDE	x	(Beginnt mit dem reservierten Wort 'FOR')
	onest	x	(Beginnt mit dem reservierten Wort 'ON')

Der Typ einer Variablen oder einer Variablengruppe kann durch ein Symbol nach dem Variablennamen angezeigt werden.

!	Reeller Typ einfacher Genauigkeit
#	Reeller Typ doppelter Genauigkeit
$	Zeichenvariable

Wenn das Typsymbol ausgelassen wird und keine Typanweisung verwendet wird, so wird die Variable als eine reelle Variable einfacher Genauigkeit angesehen. (Beziehen Sie sich für Typerklärungen auf Abschnitt 5-9-1.)

Selbst Variablen mit dem gleichen Namen werden nach ihrem Typ unterschieden. Auf diese Weise werden die nachfolgend gezeigten Variablen und Variablengruppen alle verschieden behandelt.

Variablengruppen	A!, A#, A$
Variablen	A!, A#, A$

Wenn keine Typanweisung verwendet wird, so wird die Variable A als gleich A! angesehen.

3-7-2. Variablengruppen

Eine Ansammlung von Elementen des gleichen Typs und des gleichen Namens wird als Variablengruppe bezeichnet. Die Elemente in einer Variablengruppe werden durch eine oder mehrere nicht negative ganze Zahlen oder Tiefstellungen bezeichnet. Die Anzahl der Tiefstellungen zeigt die Dimension der Variablengruppe an. Eine durch eine einzige Tiefstellung bezeichnete Variablengruppe ist eine eindimensionale Variablengruppe, und eine durch zwei Tiefstellungen bezeichnete Variablengruppe ist eine zweidimensionale Variablengruppe.

Die maximalen Werte der individuellen Tiefstellungen bestimmen die Größe der Variablengruppe. Eine Variablengruppe wird wie folgt durch die DIM-Anweisung erklärt (siehe 5-7-2).

DIM Variablengruppenname (max. Tiefstellungswert [, max. Tiefstellungswert] *)

Tiefstellungen können mit 0 oder 1 beginnen, wobei Wahl durch die Anweisung OPTION BASE erfolgt. Der Vorgabewert ist 0.

Beispiel DIM A(10, 15, 10)

Die Größe der Variablengruppe ist 11 x 16 x 11, wenn 0 durch die Anweisung OPTION BASE festgelegt ist.
Die Größe der Variablengruppe ist 11 x 15 x 10, wenn 1 durch die Anweisung OPTION BASE festgelegt ist.

Die Begrenzungen für Variablengruppen sind nachfolgend zusammengefaßt.

(1) Variablengruppendimension

Bis zu 3 Dimensionen.

(2) Tiefstellungen

Eine ganze Zahl, beginnend von Null wenn OPTION BASE gleich 0 ist.
Eine ganze Zahl, beginnend von Eins wenn OPTION BASE gleich 1 ist.

(3) Maximaler Tiefstellungswert

Keine Begrenzung (unterliegt der Speicherkapazität).

Variablengruppenelemente werden geschrieben, indem der Variablengruppenname gegeben wird, gefolgt von einer Tiefstellung oder einer Tiefstellungskette in Klammern. Die Tiefstellungen in der Kette sollten durch Kommas getrennt werden.

Variablengruppenelemente **Variablengruppenname (Tiefstellung [, Tiefstellung]*)**
Arithmetischer Ausdruck

(1) Wenn ein Ausdruck für eine Tiefstellung verwendet wird, so muß sein Wert eine ganze Zahl sein. (Jegliche Bruchteile werden vernachlässigt.)

(2) Die Anzahl der Tiefstellungen muß mit der Dimension der Variablengruppe übereinstimmen.

(3) Der Wert einer Tiefstellung muß im Bereich von 0 (oder 1) bis zum für die Variablengruppe zulässigen maximalen Wert liegen (0 oder 1 wird durch die Anweisung OPTION BASE während der Variablengruppenerklärung gewählt).

Beispiel
```
A(2*3,5)
B(1,4,10)
```

3-7-3. Handhabung von Variablen und Variablengruppen

(1) Variablen und Variablengruppen sind gemeinsam für die Programme in den Programmbereichen PROG 0 bis PROG 9.

(2) Zeichenvariablen (einschließlich Zeichenvariablenelemente) werden in einem durch die CLEAR-Anweisung bezeichneten Bereich gespeichert. Die Ausgangsgröße ist 1.023 Bytes.

(3) Der Bereich für eine einfache Variable wird reserviert, wenn sie zum erstenmal verwendet wird (implizierte Erklärung von Variablen.

(4) Wenn eine Variablengruppe zum erstenmal ohne Erklärung verwendet wird, so wird eine Variablengruppe mit allen Tiefstellungen mit einem Wert von 10 erklärt (implizierte Erklärung von Variablengruppen).

Beispiel
```
A(4,3) = 5
```
Wenn dies ohne eine DIM-Anweisung ausgeführt wird, so wird A als eine Variablengruppe von 11 x 11 angenommen. Wenn OPTION BASE 1 vorher ausgeführt wurde, so wird eine Variablengruppe von 10 x 10 angenommen.

3-8. Umwandlung von Datentypen

Wenn verschiedene Typen von Daten für die Berechnung von Ausdrücken, Zuordnungsanweisungen, Parameter in Anweisungen usw. gehandhabt werden, so werden sie automatisch entsprechend den folgenden Regeln umgewandelt:

(1) Es erfolgt keine Umwandlung zwischen numerischem Typ und Zeichentyp.

(2) Daten eines niedrigeren Genauigkeitstyps werden einfach in Daten eines höheren Genauigkeitstyps umgewandelt.

(3) Daten einer höheren Genauigkeit werden durch Vernachlässigung der niedrigsten Stellen in Daten eines niedrigeren Genauigkeitstyps umgewandelt.

3-9. Ausdrücke

Ein Ausdruck ist eine Kombination von Konstanten, Variablen, Funktionen usw., verbunden durch Operatoren, die auf einen einzigen Wert reduziert werden kann. Entsprechend dem resultierenden Datentyp werden Ausdrücke in numerische Ausdrücke und Zeichenausdrucke klassifiziert.

3-9-1. Numerische Ausdrücke

Klammern konnen auf herkömmliche Weise verwendet werden, um die Reihenfolge von Operationen zu ändern. Durch hierarchische Verwendung von Klammern können etwa sechs Operationsebenen festgelegt werden. (Die Operationsebenen andern sich entsprechend den verwendeten Funktionen, den Spezifikationen für die Variablengruppenelemente usw.)

3-9-1-1. Arithmetische Operation

Die folgenden arithmetischen Operatoren stehen zur Verfügung:

+ (Pluszeichen), – (Minuszeichen), + (Addition), – (Subtraktion), * (Multiplikation), / (Division), ∧ (Potenzierung), \ MOD (Rest einer ganzzahligen Division).

Alle diese Operatoren wirken auf Zahlen, um numerische Ergebnisse zu erhalten.

(1) + (Pluszeichen), – (Minuszeichen)

Diese Operatoren wirken auf das direkt folgende Element. Das Minuszeichen (–) multipliziert das Element mit –1, wahrend das Pluszeichen (+) keine Operation durchfuhrt. Der Datentyp verbleibt unverandert. Diese Operatoren konnen mit allen numerischen Daten verwendet werden.

Beispiel

```
+1257
-42.58%
-ALPHA
```

(2) + (Addition), – (Subtraktion), * (Multiplikation), / (Division)

Diese Operatoren wirken auf die direkt vorhergehenden und nachfolgenden Elemente. Wenn diese Elemente von verschiedenen Datentypen sind, so wird das Element mit geringerem Genauigkeitstyp vor der Operation in ein Element mit hoherem Genauigkeitstyp umgewandelt. Das Ergebnis ist vom höheren Genauigkeitstyp.

Wenn der absoluter Wert den Bereich des Datentyps überschreitet (z.B. 10^{100}), so tritt ein Überlauffehler ein. Wenn der absolute Wert des Ergebnisses kleiner als 10^{-99} ist, so wird er als 0 angesehen.

Division eines Wertes durch Null resultiert in einem mathematischen Fehler.

Beispiel

```
12.5+4          (=16.5)
0.5/1E99        (=0)
```

(3) ∧ (Potenzierung)

X ∧ Z erzeugt das Ergebnis X^Y. Das Ergebnis ist ein reeller Typ mit entweder einfacher oder doppelter Genauigkeit. Wenn X negativ ist und Y nicht eine ganze Zahl ist, so tritt ein Fehler ein. Wenn X und Y beide 0 sind (0 ∧ 0), so tritt ein Fehler ein.

Potenzieren von 0 mit einem negativen Wert erzeugt auch einen Fehler.

Die anderen Regeln für arithmetische Operationen gelten auch für Potenzierung.

Beispiel

```
-2∧4          (=-16)
(-2)∧4        (=  16)
(-2)∧1.2      (Fehler)
```

(4) MOD (Rest einer ganzzahligen Division)

Diese Operatoren wirken auf die direkt vorhergehenden und folgenden Elemente. Beide Elemente werden zu einem ganzzahligen Typ umgewandelt. Der absolute Wert des ersten Elements wird durch den absoluten Wert des Zweiten Elements geteilt, und das Ergebnis besteht aus dem Rest mit dem Vorzeichen des ersten Elements. Ein Fehler tritt auf, wenn das zweite Element (der Divisor) 0 ist.

Beispiel

```
-15 MOD 7          (=-1)
1.23E3 MOD 3.45E2  (=192)
```

3-9-1-2. Vergleichsoperator

Die folgenden Vergleichsoperatoren stehen zur Verfügung:

= (gleich), <>, >< (ungleich), < (kleiner als), > (größer als), =<, <= (nicht größer als), >=, => (nicht kleiner als).

Eine Vergleichsoperation ist nur gültig, wenn beide Elemente vom Zeichentyp sind oder wenn beide Elemente vom numerischen Typ sind. Wenn die Elemente im Fall von numerischen Elementen von verschiedener Genauigkeit sind, so wird der niedrigere Genauigkeitstyp vor dem Vergleich zum höheren Genauigkeitstyp umgewandelt. Wenn der absolute Wert des Unterschieds kleiner als 1E-99 ist, so werden die Elemente als gleich angesehen.

Im Fall von Elementen vom Zeichentyp werden die internen Kode beider Elemente der Reihe nach von links nach rechts verglichen. Wenn die beiden Elemente eine unterschiedliche Zeichenkettenlänge haben, so wird der Vergleich so weit wie möglich durchgeführt (die Länge des kürzeren Elementes). Wenn in diesem Fall beide Elemente gleich sind, so wird die kürzere Zeichenkette als kleiner angenommen.

Das Ergebnis einer Vergleichsoperation ist vom ganzzahligen Typ: Wenn das Ergebnis "wahr" ist, so ist der Wert -1 (&HFFFF), und wenn das Ergebnis "falsch" ist, so ist der Wert 0.

Beispiel

```
125 > 12          (=-1)
"DEE" < "ABCD"    (= 0)
"ABCD" = "ABC"    (= 0)
```

3-9-1-3. Logischer Operator

Die folgenden logischen Operatoren stehen zur Verfügung:

NOT (Verneinung), AND (logisches Produkt), OR (logische Summe) und XOR (exklusive logische Summe).

Diese Operatoren wirken auf ganzzahlige Elemente und erzeugen ein ganzzahliges Ergebnis. Wenn die Elemente vom reellen Typ sind, so werden sie für die Operation zu ganzzahligen Typen umgewandelt. Im Fall eines ganzzahligen Typs wird der interne Kode durch ein 16-Bit-Komplement von 2 dargestellt, und die Operation wird für jedes Bit durchgeführt.

(1) NOT (Verneinung)

Jedes Bit des auf diesen Operator folgenden Elements wird umgekehrt.

NOT

X	NOT X
0	1
1	0

(2) AND (logisches Produkt), OR (logische Summe), XOR (exklusive logische Summe)

Bitweise Operationen für die Elemente vor und nach dem Operator werden wie folgt durchgeführt:

AND

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Andere logische Operationen konnen durch Kombination dieser Operatoren durchgeführt werden. Zum Beispiel:

Implikation X IMP Y —— (NOT X) OR Y

Äquivalenz X EQV Y —— (NOT X XOR Y)

IMPLICATION

X	Y	X IMP Y
0	0	1
0	1	1
1	0	0
1	1	1

EQUIVALENCE

X	Y	X EQV Y
0	0	1
0	1	0
1	0	0
1	1	1

3-9-1-4. Bewertung von numerischen Ausdrücken und Vorrang von Operationen

Für Operationen werden die folgenden Vorrangzusammenhänge verwendet. Wenn zwei oder mehr Operationen die gleiche Vorrangebene haben, so werden diese von links nach rechts ausgeführt.

1. Elemente in Klammern
2. Funktionen
3. Exponenten
4. Pluszeichen (+) und Minuszeichen (–)
5. *, /, MOD
6. + und –
7. Vergleichsoperatoren
8. NOT
9. AND
10. OR und XOR

Weiterhin gelten die folgenden Regeln:

(1) Wenn ein Fehler während einer Operation eintritt, so wird die Operation sofort beendet.

(2) Wenn ein Ausdruck mit NOT beginnt, so sollte dies in Klammern eingeschlossen sein.

(3) Wenn einem NOT ein weiteres NOT folgt, so muß das zweite NOT in Klammern eingeschlossen sein.

Beispiel NOT (NOT A)

(4) Da die Vergleichsoperatoren den gleichen Vorrang haben, sollten Klammern verwendet werden, um den Vorrang unter ihnen zu klären.

Beispiel 25 > 3 + 5 > verursacht einen Fehler.
Ändern Sie die Schreibart zu (25 > 3 + 5) > 17.

3-9-2. Zeichenausdrücke

+ ist der einzige Operator, der mit Zeichendaten verwendet werden kann. Dieser Operator verkettet das direkt vorhergehende und das direkt folgende Element.

Beispiel "ABC" + "DEF" + "G" (= "ABCDEFG")

Die Länge der Zeichenkette darf niemals 255 Zeichen überschreiten. Beachten Sie weiterhin, daß im Ausdruck keine Klammern verwendet werden können.

3-10. Funktionen

Funktionen werden wie folgt geschrieben.

Funktionsname [(Argument [, Argument] *)]

Funktionen werden entsprechend dem Typ des Wertes, den sie annehmen, in numerische Funktionen und Zeichenfunktionen klassifiziert. Sie können auch entsprechend der Definitionsmethode in eingebaute Funktionen und durch den Benutzer definierte Funktionen klassifiziert werden. Eine eingebaute Funktion ist eine Funktion, deren Name und Bedeutung vorherbestimmt sind (z.B. sin und cos). Eine durch den Benutzer definierte Funktion wird in einem Programm definiert, und es wird durch ihren Namen Bezug auf sie genommen.

3-10-1. Durch den Benutzer definierte Funktionen (FN-Funktionen)

Eine durch den Benutzer definierte Funktion wird durch die DEFFN-Anweisung definiert (siehe 5-12-1).

DEF FN Funktionsname [(Formalargument [, Formalargument] *)] = Ausdruck

Formalargumente werden verwendet, um das Format des Ausdrucks zu definieren. Bezug auf die durch den Benutzer definierte Funktion erfolgt im folgenden Format (siehe 5-12-2):

FN Funktionsname [(Argument [, Argument*)]

3-10-2. Namen von durch den Benutzer definierten Funktionen

Die Namen von durch den Benutzer definierten Funktionen müssen den gleichen Regeln wie für Variablennamen folgen. Dem Namen einer durch den Benutzer definierten Funktion kann ein Symbol zur Anzeige seines Typs nachgestellt werden.

3-11. In diesem Handbuch verwendete Notierung

Zur Beschreibung der Satzlehre werden in diesem Handbuch die folgenden Notierungen verwendet:

- Fettgedruckte Worte — Worte, die so wie sie erscheinen geschrieben werden müssen.
- $\{^{\text{XXXX}}_{\text{oooo}}\}$ — Eins der Elemente in der Klammer muß eingeschlossen sein.
- [ooo] — Eins der Elemente in der Klammer kann ausgelassen werden.
- ooo* — Ein Element mit einem Stern (*) an der rechten Seite kann wiederholt erscheinen.

Beispiel **DATA** [data] [, [data]] *

Da jedes Element in eckigen Klammern eingeschlossen ist, kann einfach "DATA" geschrieben werden. Das Element 'data' kann wiederholt werden, da dieses Element einen Stern (*) an der rechten Seite hat. Es kann also "DATA data, data" geschrieben werden.

RESTORE $\left[\left\{\begin{array}{l}\text{Zeilennummer}\\ \text{(numerischer Ausdruck)}\end{array}\right\}\right]$

In diesem Fall ist einer der folgenden drei Ausdrücke möglich:
1) RESTORE
2) RESTORE Zeilennummer
3) RESTORE (numerischer Ausdruck)

4. BEFEHLE

Der Begriff 'Befehl' wird normalerweise für eine in Direktbetriebsart verwendete Anweisung benutzt.

4-1. Systembefehle

4-1-1. PROG-Befehl

PROG	Programmbereichsnummer (0 bis 9)

Funktion

Wahl des Programmbereichs.

Parameter

(1) Programmbereichsnummer 0 bis 9.

Erklärung

(1) Wahl des festgelegten Programmbereichs.

(2) Nach Ausführung dieses Befehls wartet das System auf Eingabe des nächsten Befehls.

(3) Alle offenen Dateien werden geschlossen.

(4) Alle durch den Benutzer definierten Funktionen werden gelöscht.

(5) Der gegenwärtige Programmbereich bleibt auch bei Ausschalten der Stromversorgung wirksam.

Beispiel

PROG 3

In Zusammenhang stehende Punkte

GOTO GOSUB

4-1-2. PASS-Befehl

PASS Kennwort / Zeichenkonstante

Funktion

Einstellen oder Entfernen eines Kennwortes.

Parameter

(1) Ein Kennwort ist eine aus 1 bis 8 Zeichen bestehende Zeichenkonstante. Das Anführungszeichen an der rechten Seite kann nicht ausgelassen werden.

Erklärung

(1) Wenn dieser Befehl ausgeführt wird, wenn kein Kennwort bezeichnet worden ist, so wird das in diesem Befehl bezeichnete Kennwert eingestellt.

Wenn dieser Befehl ausgeführt wird, wenn ein Kennwort bezeichnet worden ist, so wird das Kennwort zurückgestellt, wenn das gegenwartig eingestellte Kennwort dem in diesem Befehl bezeichneten Kennwort entspricht. Wenn beide Kennworte nicht übereinstimmen, so tritt ein PR-Fehler ein.

(2) Ein Kennwort ist eine aus 1 bis 8 Zeichen bestehende Zeichenkonstante. Für das Kennwort konnen Groß- und Kleinbuchstaben, grafische Symbole, Leerstellen usw. verwendet werden. Das Anführungszeichen (") kann nicht in einem Kennwort verwendet werden. (Die internen Kode 20, 21, 23-7E, 80-9F und E0-FE können für das Kennwort verwendet werden.)

(3) Es kann nur ein Kennwort für den gesamten Programmbereich verwendet werden.

(4) Wenn ein Kennwort eingestellt worden ist, so konnen die folgenden Befehle und Anweisungen nicht in Direktbetriebsart ausgeführt werden:

LIST, LLIST, EDIT, RENUM, CALL, SAVE, LOAD, TRON, POKE.

(5) Wenn ein Kennwort eingestellt wird, so wird der Bildschirm gelöscht.

(6) Dieser Befehl kann nicht in einem Programm verwendet werden.

(7) Das Kennwort bleibt auch bei Ausschalten der Stromversorgung gültig.

Beispiel

```
PASS "ORG"
```

In Zusammenhang stehende Punkte

Verwenden Sie die SAVE-Anweisung, um einem zu speichernden Programm ein Kennwort zuzuordnen.
Ein Kennwort für eine aubere Datei wird durch Ausführung der LOAD-Anweisung eingestellt. Wenn ein Programm mit einem Kennwort ein anderes Programm mit einem anderen Kennwort abruft, so wird das Kennwort des abgerufenen Programms wirksam.
SAVE
LOAD

NEW [ALL]

Funktion

NEW löscht ein Programm, und NEW [ALL] löscht alle Programme und Variablen.

Parameter

(1) ALL (wenn bezeichnet) löscht alle Programme in PROG 0 bis PROG 9 und alle Variablen. Wenn ALL nicht bezeichnet ist, so wird das Programm im gegenwärtig bezeichneten Programmbereich gelöscht.

Erklärung

I. Wenn ALL bezeichnet ist:

(1) Dieser Befehl stellt alle Kennworte zurück und löscht alle Programme und Variablen.

(2) Nach Ausführung dieses Befehls zeigt das System die Bereitschaftsnachricht(READY) und wartet auf Eingabe eines Befehls.

(3) Spezifikationen von Variablentypen und Definitionen von durch den Benutzer definierten Funktionen werden gelöscht.

(4) Jegliche in der Durchführung befindliche Datenverarbeitung wird unterbrochen, und die Anzahl der Dateipuffer wird 0. Aus diesem Grund wird der Inhalt einer offenen Datei nicht garantiert.

II. Wenn ALL nicht bezeichnet ist:

(1) Dieser Befehl löscht das Programm im gegenwärtig bezeichneten Programmbereich, aber er löscht nicht die Variablen.

(2) Dieser Befehl kann nicht ausgeführt werden, wenn ein Kennwort eingestellt worden ist.

(3) Nach Ausführung dieses Befehls wartet das System auf den nächsten Befehl.

(4) Die Definitionen von durch den Benutzer definierten Funktionen werden gelöscht.

(5) Gegenwärtig offene Dateien werden geschlossen.

Beispiel

```
NEW
NEW ALL
```

SYSTEM

Funktion

Anzeige des Programmbereichzustands bzw. des Dateibereichszustands (CETL).

Erklärung

(1) Wenn der Betriebsartenschalter auf BASIC gestellt ist:

Durch diesen Befehl werden die Größe des nicht verwendeten (verbleibenden) Programmbereichs, die Größen der durch PROG 0 bis PROG 9 verwendeten Bereiche und die Dateigröße in Bytes angezeigt.

```
>SYSTEM
12550 Bytes Free

0:   214
3:   1426

Ready P0
>
```

(2) Wenn der Betriebsartenschalter auf CETL gestellt ist:

Durch diesen Befehl werden die Größe des nicht verwendeten Dateibereichs, die Größen der durch FILE 0 bis FILE 9 verwendeten Bereiche und die Programmgröße in Bytes angezeigt.

```
>SYSTEM
14291 Bytes Free

0:   57
1:   863

Ready F1
>
```

(3) Nach der Ausführung dieses Befehls wartet das System auf den nächsten Befehl.

4-1-5. KEY-Befehl

KEY	Funktionstastennummer 0 bis 9	Tasteninhalt Zeichenausdruck

Funktion

Definition einer programmierbaren Funktionstaste.

Parameter

(1) Funktionstastennummer
Eine der Ziffern von 0 bis 9.

(2) Tasteninhalt

Ein aus bis zu 15 Zeichen bestehender Zeichenausdruck.

Erklärung

(1) Dieser Befehl definiert den Inhalt einer der programmierbaren Funktionstasten (PF-Taste) 0 bis 9. Nach Ausführung dieses Befehls führt die PF-Taste die gleiche Operation aus, als ob die enthaltenen Zeichen einzeln eingegeben worden wären.

(2) Der Zeichenausdruck kann einen Steuerkode (01 bis 1F) enthalten. Beziehen Sie sich für den Zusammenhang zwischen Koden und Tasten auf die Tasten-/Kode-Tabelle am Ende dieses Handbuchs.

(3) Der Zeichenausdruck kann eine Länge von bis zu 15 Zeichen haben. Zeichen, die diese Länge überschreiten, werden nicht beachtet.

(4) Zu Anfang und nach Durchführung des RESET-Befehls sind die PF-Tasten wie folgt definiert:

BASIC-Betriebsart

```
0:EDIT
1:PROG
2:SYSTEM
3:LIST
4:RUN
5:
6:FILES
7:LOAD"
8:SAVE"
9:P.DATE$,TIME$
```

CETL-Betriebsart

```
0:
1:FILE
2:SYSTEM
3:
4:P.FRE
5:
6:FL(
7:RC(
8:IT(
9:P.DATE$,TIME$
```

(5) Die Tastendefinitionsbetriebsart mit dem KEY-Befehl bleibt auch bei Ausschalten der Stromversorgung wirksam.

Beispiel

(1) KEY3, "LIST" + CHR$(13)

Dieser Befehl führt die gleiche Funktion aus wie **L I S T RETURN.**

(2) KEY 1, "RUN" † CHS(13) + "RUN" + CHRS(13)

Dieser Befehl führt die gleiche Funktion aus wie **R U N RETURN R U N RETURN**. Durch Druck auf die Taste PF1 wird also das Programm zweimal ausgeführt.

In Zusammenhang stehender Punkt

Tasten-/Kode-Tabelle
KEY LIST

KEY LIST

Funktion

Anzeige der Inhalte der definierten programmierbaren Funktionstasten (PF-Tasten).

Erklärung

(1) Dieser Befehl zeigt die Inhalte der definierten programmierbaren Funktionstasten an.

BASIC-Betriebsart

```
0:EDIT
1:PROG
2:SYSTEM
3:LIST
4:RUN
5:
6:FILES
7:LOAD"
8:SAVE"
9:P.DATE$,TIME$
```

CETL-Betriebsart

```
0:
1:FILE
2:SYSTEM
3:
4:P.FRE
5:
6:FL
7:RC
8:IT
9:P.DATE$,TIME$
```

Die Steuerkode werden wie folgt in der Form von Zeichen angezeigt.

		08		10		18	
01		09		11		19	
02		0A		12		1A	
03		0B		13		1B	
04		0C		14		1C	
05		0D		15		1D	
06		0E		16		1E	
07		0F		17		1F	

(2) Nach Ausführung dieses Befehls wartet das System auf den nachsten Befehl.

Beispiel

KEY LIST

Siehe in diesem Zusammenhang auch

KEY

4-1-7. CLEAR-Befehl

CLEAR [Zeichenbereichsgröße] , [BASIC-Bereichsobergrenze]

numerischer Ausdruck — numerischer Ausdruck

Funktion

Löschen aller Variablen, Funktionen und durch den Benutzer definierten Funktionen, und Festlegung von Zeichenbereichsgröße und BASIC-Bereichsobergrenze.

Parameter

(1) Zeichenbereichsgröße

Ein zu einer ganzen Zahl bewerteter numerischer Ausdruck. Wenn dieser Parameter ausgelassen wird, so wird die gegenwärtige Zeichenbereichsgröße festgelegt.

$0 \leqslant$ Zeichenbereichsgröße $\leqslant$ Speicherbereich

(2) BASIC-Bereichsobergrenze

Ein zu einer ganzen Zahl bewerteter numerischer Ausdruck. Wenn dieser Parameter ausgelassen wird, so wird die gegenwärtige BASIC-Bereichsobergrenze festgelegt.

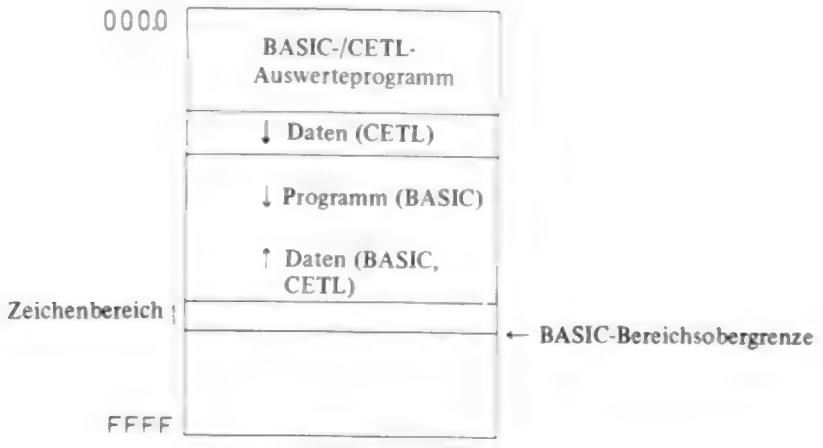
Speicherbereichobergrenze $\leqslant$ BASIC-Bereichobergrenze $\leqslant 65534$

Erklärung

(1) Dieser Befehl löscht alle Variablen, Funktionen und durch den Benutzer definierten Funktionen, und legt die Zeichenbereichsgröße und die BASIC-Bereichsobergrenze fest. Er löscht weiterhin Typspezifikationen durch DEF-Anweisungen.

(2) Dieser Befehl kann nicht für die Anweisungen FOR ~NEXT und WHILE ~WEND oder für Unterprogramme verwendet werden.

(3) Der Zeichenbereich enthält alle Werte von Zeichenvariablen. Zu Anfang und nach Durchführung des RESET-Befehls ist die Große dieses Bereiches 1.023 Bytes. Um Zeichendaten über diese Größe hinaus handhaben zu können, muß ein größerer Zeichenbereich durch den CLEAR-Befehl festgelegt werden.



(4) Die anfängliche Obergrenze des BASIC-Bereichs ist die Adresse FFFE.

Siehe in diesem Zusammenhang auch

AREA
MOUNT

4-1-8. RESET-Befehl

RESET

Funktion

Initialisieren des Direktzugriffsspeichers des Haupgerätes.

Erklärung

(1) Dieser Befehl initialisiert den Direktzugriffsspeicher des Hauptgerätes.

(2) Die initialisierten Punkte und die Ergebnisse sind wie folgt:

DEF-Typ:	Einfache Genauigkeit
OPTION BASE:	0
ANGLE:	0
Zufallszahlenmatrix:	Zufallszahl
Stapel:	Numerischer Stapel, FOR-, GOSUB-Stapel
Variablen:	Alle gelöscht.
Zeichenbereich:	1023 Bytes
BASIC-Obergrenze:	65534
BASIC-Bereich:	14189 Bytes. Gesamtes Programm gelöscht. PROG 0 ist eingestellt.
CETL-Bereich:	15212 Bytes. Gesamte Datei gelöscht. FILE 0 ist eingestellt.
Dateipuffer:	F0. 316 Bytes gelöscht.
Tasteninhalt (KEY):	Siehe KEY-LIST-Befehl.
Statistischer Wert:	
Kennwort:	Freigegeben.
Absolute Koordinaten:	Ursprung (0, 0). X- und Y-Zuwachs sind auf 1 eingestellt.
Verfolgungsbetriebsart:	Aus.
Durch den Benutzer definierte Funktionen:	Gelöscht.

(3) Nach Ausführung der RESET-Anweisung wird die Anfangsnachricht für die BASIC-Betriebsart bzw. für die CETL-Betriebsart angezeigt, und das System wartet auf den nächsten Befehl.

* Nach dem Auswechseln von Batterien (einschließlich der Speicherschutz-Batterien) muß Initialisierung mit diesem Befehl durchgeführt werden.

4-1-9. AREA-Befehl

AREA CETL-Bereichsgröße
Numerischer Ausdruck

Funktion

Festlegen der Größe des CETL-Bereichs.

Parameter

CETL-Bereichsgröße	Numerischer Ausdruck 0 ⩽ CETL-Bereichsgröße ⩽ 30424

Erklärung

(1) Einstellen der CETL-Bereichsgröße.

(2) Wenn schon Dateien im CETL-Bereich vorhanden sind, so kann die CETL-Bereichsgroße nicht auf eine kleinere Große als für diese Dateien erforderlich eingestellt werden.

(3) Die CETL-Bereichsgroße wird durch Ausführung des AREA-Befehls geandert, und der BASIC-Bereich wird entsprechend der Änderungsbetriebsart bewegt.

(4) Die Größe der durch MOUNT reservierten Dateipuffer ist jeweils 316 Bytes. Dieser Bereich wird aus dem CETL-Bereich gewonnen.

(5) Nach Ausführung des AREA-Befehls wird CLS durchgeführt, die Anfangsnachricht wird angezeigt, und das System wartet auf den nächsten Befehl.

Siehe in diesem Zusammenhang auch

MOUNT, RESET, CLEAR

4-2. Programm redigieren

4-2-1. RENUM-Befehl

RENUM	[Neue Zeilennumer] Zeilennummer	[,[Alte Zeilennummer] Zeilennummer	[,[Zuwachs] Zeilennummer

Funktion

Änderung der Zeilennummern in einem bestimmten Intervall.

Parameter

(1) Neue Zeilennummer

Die erste Zeilennummer nach der Neunummerierung.
Der Vorgabewert für diesen Parameter ist 10.

(2) Alte Zeilennummer

Die erste zu ändernde Zeilennummer.
Wenn dieser Parameter ausgelassen wird, so wird die erste Zeile des Programms bezeichnet.

(3) Zuwachs

Der Zuwachs, mit dem die Zeilennummern zu ändern sind.
Der Vorgabewert für diesen Parameter ist 10.

Die neue Zeilennummer, die alte Zeilennummer und der Zuwachs sollten jeweils im Bereich von 1 bis 64999 sein.

Erklärung

(1) Dieser Befehl andert die Programmzeilennummern, beginnend mit der bezeichneten alten Zeilennummer, zu neuen Zeilennummern, beginnend mit der festgelegten neuen Zeilennummer, mit dem festgelegten Zuwachs. In diesem Fall werden die in GOTO-Anweisungen usw. erscheinenden Zeilennummern im Programm auch geändert. Wenn ein Satzlehrefehler eintritt, so kann dieser Befehl nicht korrekt ausgeführt werden.

(2) Wenn das Programm eine illegale Zeilennummer enthält, so wird dieser Befehl nicht ausgeführt, und es tritt ein UL-Fehler auf.

(3) Wenn die durch die alte Zeilennummer bezeichnete Zeile nicht im Programm vorhanden ist, so tritt ein Fehler auf.

(4) Wenn die bezeichnete alte Zeilennummer nicht die erste Zeilennummer des Programms ist, so muß die bezeichnete neue Zeilennummer größer als die Zeilennummer direkt vor der bezeichneten alten Zeilennummer sein.

(5) Wenn als Ergebnis der Neunumerierung eine Zeilennummer über 64999 erscheint, so tritt ein Fehler auf.

(6) Nach der Ausführung dieses Befehls wartet das System auf den nächsten Befehl.

(7) Dieser Befehl kann nicht ausgeführt werden, wenn ein Kennwort eingestellt worden ist.

Beispiel

```
 10 FOR I=1 TO 10
 22 PRINT SQR(I)
 30 NEXT
 40 FOR J=1 TO 100
 50 PRINT LOG(J)
160 NEXT J
```

```
Ready P0
RENUM

Ready P0
LIST
    10 FOR I=1 TO 10
    20 PRINT SQR(I)
    30 NEXT
    40 FOR J=1 TO 100
    50 PRINT LOG(I)
    60 NEXT J
```

```
RENUM100,10,15

Ready P0
LIST
   100 FOR I=1 TO 10
   115 PRINT SQR(I)
   130 NEXT
   145 FOR J=1 TO 100
   160 PRINT LOG(I)
   175 NEXT J
```

4-2-2. LIST-Befehl

LIST [Ausgangszeilennummer] [{,} {-} [Endzeilennummer]]

Ausgangszeilennummer: Zeilennummer oder "."

Endzeilennummer: Zeilennummer oder "."

Funktion

Anzeige des Inhalts eines Programms auf der Flüssigkristallanzeige.

Parameter

(1) Ausgangszeilennummer
- Die erste anzuzeigende Zeilennummer.
- Wenn dieser Parameter ausgelassen wird, so wird der Anfang des Programms bezeichnet.

(2) Endzeilennummer
- Die letzte anzuzeigende Zeilennummer.
- Wenn dieser Parameter ausgelassen wird, so wird das Ende des Programms bezeichnet.

Die Ausgangszeilennummer und die Endzeilennummer mussen im Bereich von 1 bis 64999 sein.

Erklärung

(1) Dieser Befehl zeigt die bezeichneten Zeilen des Programms auf dem Bildschirm an.

(2) "," oder "–" kann verwendet werden, um die Zeilennummern zu trennen. Beide Zeichen haben die gleiche Wirkung.

(3) Der Bereich der Anzeige kann auf eine der folgenden fünf Arten bezeichnet werden:

1)	Das gesamte Programm:	LIST
2)	Nur die bezeichnete Zeile:	LIST Zeilennummer
3)	Zwischen zwei bezeichneten Zeilen:	LIST Ausgangszeilennummer , Endzeilennummer
4)	Vom Anfang zur bezeichneten Zeile:	LIST , Endzeilennummer
5)	Von der bezeichneten Zeile zum Ende:	LIST Ausgangszeilennummer

(4) Wenn die als Ausgangszeilennummer bezeichnete Zeilennummer nicht existiert, so wird die nächstgrößere Zeilennummer als die Ausgangszeilennummer genommen.

(5) Wenn die als Endzeilennummer bezeichnete Zeilennummer nicht existiert, so wird die größte Zeilennummer, die diese Nummer nicht überschreitet, verwendet.

(6) Die Ausgangszeilennummer darf nicht größer als die Endzeilennummer sein.

(7) Der LIST-Betrieb kann durch Druck auf die Taste STOP/CONT unterbrochen und durch erneuten Druck auf diese Taste wieder aufgenommen werden.

(8) Nach der Ausführung dieses Befehls wartet das System auf den nächsten Befehl.

(9) Dieser Befehl kann nicht ausgeführt werden, wenn ein Kennwort für den gegenwärtig bezeichneten Programmbereich oder für den gesamten Programmbereich eingestellt worden ist.

(10) Wenn die Länge der angezeigten Zeile 255 Zeilen überschreitet, so tritt ein Fehler auf und der LIST-Betrieb wird abgebrochen.

(11) Wenn dieser Befehl ausgeführt wird, so werden offene Dateien geschlossen.

Beispiel

```
    10 FOR I=1 TO 200
    20 PRINT I,SQR(I)
    30 NEXT I
    40 FOR J=1 TO 100
0:PRINT SQR(J+3):NEX
T
    50 PRINT "END"
    60 END
```

```
LIST ,40
    10 FOR I=1 TO 200
    20 PRINT I,SQR(I)
    30 NEXT I
    40 FOR J=1 TO 100
0:PRINT SQR(J+3):NEX
T
```

```
LIST 20,55
    20 PRINT I,SQR(I)
    30 NEXT I
    40 FOR J=1 TO 100
0:PRINT SQR(J+3):NEX
T
    50 PRINT "END"
```

```
LIST 40,
    40 FOR J=1 TO 100
0:PRINT SQR(J+3):NEX
T
    50 PRINT "END"
    60 END
```

Siehe in diesem Zusammenhang auch

LLIST

4-2-3. EDIT-Befehl

EDIT [Zeilennummer]

Funktion

Anzeige der festgelegten Zeile auf dem Bildschirm und Einleitung der Redigierbetriebsart.

Parameter

(1) Zeilennummer

Die Nummer der anzuzeigenden Zeile. Wenn dieser Parameter ausgelassen wird, so wird die erste Zeile des Programms angezeigt. Die Zeilennummer muß im Bereich von 1 bis 64999 sein.

Erklärung

(1) Durch diesen Befehl wird die bezeichnete Programmzeile auf dem Bildschirm angezeigt, und es wird Redigierbetrieb eingeleitet.

(2) Der Positionsanzeiger bewegt sich zu der auf das Ende der Zeile folgenden Position. Wenn die Zeile jedoch 180 Zeichen oder mehr hat, so werden nur die ersten 180 Zeichen angezeigt, und der Positionsanzeiger bewegt sich zur rechten unteren Ecke des Bildschirms.

(3) Durch Druck auf die Rücklauftaste im Redigierbetrieb wird der Redigierprozeß abgeschlossen, und die modifizierte Zeile bleibt erhalten.

(4) Der Redigierbetrieb kann durch Druck auf die Abbruchtaste (BREAK) oder die Rücklauftaste (RETURN) beendet werden.

(5) Wenn die Taste CLS oder die Taste HOME während des Redigierbetriebs gedrückt wird, so wird der Inhalt des Bildschirms nicht modifiziert, wenn anschließend die Rücklauftaste gedrückt wird.

(6) Wenn die bezeichnete Zeilennummer nicht existiert, so wird die nächsthöhere Zeilennummer angezeigt. Wenn eine höhere als die letzte Zeilennummer bezeichnet wird, so kann der Redigierbetrieb nicht eingeleitet werden.

(7) Der EDIT-Befehl kann nicht ausgeführt werden, wenn ein Kennwort eingestellt ist, oder wenn der EDIT-Befehl in einem Programm enthalten ist und BASIC von der CETL-Betriebsart aufgerufen wird.

(8) Die Bewegung des Positionsanzeigers im Redigierbetrieb ist die gleiche wie bei normalem Betrieb: Zeichen (Zeichenkode 20 bis 7E und 8Q bis FE) können nicht eingegeben werden, wenn die Position des Positionsanzeigers über der logischen Zeile ist. Wenn das Ende der logischen Zeile auf dem Bildschirm ist, so kann die Taste ↓ nicht in der untersten Zeile und die Taste → nicht am rechten Ende der Zeile verwendet werden.
(Das gleiche gilt, wenn der INPUT-Befehl ausgeführt wird oder wenn das Programm durch den STOP-Befehl oder die Taste STOP angehalten worden ist.)

4-3. Ausführungssteuerbefehle

4-3-1. RUN-Befehl

RUN [Ausführungsanfangszeile]

Zeilennummer

Funktion

Ausführung eines Programms.

Parameter

(1) Ausführungsanfangszeile
- Eine Zeilennummer im Bereich von 1 bis 64999.
- Wenn dieser Parameter ausgelassen wird, so wird die Programmanfangszeile bezeichnet.

Erklärung

(1) Durch diesen Befehl wird das Programm von der bezeichneten Ausführungsanfangszeile an bzw. bei Auslassen des Parameters vom Anfang des Programms an ausgeführt.

(2) Wenn die bezeichnete Zeilennummer nicht existiert, so kann dieser Befehl nicht ausgeführt werden.

(3) Wenn dieser Befehl ausgeführt wird, so werden offene Dateien geschlossen.

(4) Dieser Befehl hebt vorherige DEF-Typspezifikationen auf.

(5) Alle durch den Benutzer definierten Funktionen werden gelöscht.

(6) Durch diesen Befehl wird der OPTION-BASE-Wert auf 0 gestellt.

(7) Dieser Befehl initialisiert den Zufallszahlengenerator.

(8) Variablen und Variablengruppen werden nicht gelöscht.

Beispiel

```
RUN 100
```

Siehe in diesem Zusammenhang auch

RUN, LOAD

TR ON

Funktion

Verfolgung der anschließenden Programmausführung.

Erklärung

(1) Dieser Befehl schaltet auf Verfolgungsbetriebsart um. In der Verfolgungsbetriebsart werden die Zeilennummern, die ausgeführt werden, zusammen mit der Programmbereichsnummer auf dem Bildschirm angezeigt.

(2) Die Zeilennummer wird angezeigt, wenn die erste Anweisung der Zeile ausgeführt wird.

(3) Die Verfolgungsbetriebsart bleibt geltend, bis ein TROFF-Befehl oder ein RESET-Befehl ausgeführt wird.

Beispiel

```
10 S=0
20 FOR I=1 TO 10
30 S=S+I
40 NEXT
50 PRINT S
60 END
```

```
RUN
[0:10] [0:20] [0:30]
[0:40] [0:30] [0:40
] [0:30] [0:40] [0:3
0] [0:40] [0:30] [0:
40] [0:30] [0:40] [0
:30] [0:40] [0:30] [
0:40] [0:30] [0:40]
[0:30] [0:40] [0:50]
  55
[0:60]
```

4-3-3. TROFF-Befehl

TROFF

Funktion

Verfolgung der Programmausführung wird zeitweilig unterbrochen.

Erklärung

(1) Die durch TRON eingeschaltete Verfolgungsbetriebsart wird ausgeschaltet.

Beispiel

```
10 TRON
20 FOR I=1 TO 10
30 S=S+I
40 NEXT
50 TROFF
60 A=SIN(30)
70 PRINT A
80 END
```

```
RUN
[0:20] [0:30] [0:40]
[0:30] [0:40] [0:30
] [0:40] [0:30] [0:4
0] [0:30] [0:40] [0:
30] [0:40] [0:30] [0
:40] [0:30] [0:40] [
0:30] [0:40] [0:30]
 55
```

5. GRUNDLEGENDE OPERATIONEN UND FUNKTIONEN

5-1. Ausführungssteuerung

5-1-1. END-Anweisung

```
END
```

Funktion

Beendigung der Programmausführung.

Erklärung

(1) Diese Anweisung beendet die Programmausführung, und das System wartet auf den nächsten Befehl.

(2) Wenn diese Anweisung ausgeführt wird, so werden offene Dateien geschlossen.

5-1-2. STOP-Anweisung

```
STOP
```

Funktion

Zeitweilige Unterbrechung der Programmausführung.

Erklärung

(1) Diese Anweisung unterbricht die Programmausführung zeitweilig. Das System wartet auf den nachsten Befehl. Die Ausführung des Programms wird wieder aufgenommen, wenn die Taste STOP/CONT gedrückt wird oder wenn ein CONT-Befehl in Direktbetriebsart ausgeführt wird.

Beispiel

```
10 PRINT "**** START ****"
20 STOP
30 PRINT "**** CONTINUE ****"
40 END
```

```
RUN
**** START ****

Stop P1 in 20
>**** CONTINUE ***

Ready P1
```

5-1-3. GOTO-Anweisung

```
GOTO { Zielzeilennummer
       Zeilennummer
       PROG Programmbereichsnummer }
```

Programmbereichsnummer: 0 bis 9

Funktion

Verursacht einen unbedingten Sprung zum bezeichneten Ziel.

Parameter

(1) Zielzeilennummer: Eine Zeilennummer von 1 bis 64999.

(2) Programmbereichsnummer: Eine Ziffer von 0 bis 9.

Erklärung

(1) Diese Anweisung verursacht einen unbedingten Sprung zum bezeichneten Ziel.

(2) Wenn das Ziel durch die Zeilennummer bezeichnet wird, so wird ein Sprung zur bezeichneten Zeilennummer des gegenwärtigen Programmbereichs durchgeführt.

(3) Wenn das Ziel durch eine Programmbereichsnummer bezeichnet ist, so wird ein Sprung zur ersten Zeile des bezeichneten Programmbereichs durchgeführt.

Beispiel

```
 10 PRINT "START"
 20 GOTO 100
 30 PRINT "LINE 30"
 40 END
100 PRINT "LINE 100"
110 END
```

```
RUN
START
LINE 100
```

5-1-4. GOSUB/RETURN-Anweisung

5-1-4-1. GOSUB-Anweisung

GOSUB { Zielzeilennummer / Zeilennummer / PROG Programmbereichsnummer }

0 bis 9

Funktion

Verursacht einen Unterprogrammsprung zum bezeichneten Ziel.

Parameter

(1) Zielzeilennummer: Eine Zeilennummer von 1 bis 64999.

(2) Programmbereichsnummer: Eine Ziffer von 0 bis 9.

Erklärung

(1) Diese Anweisung verursacht einen Unterprogrammsprung zum bezeichneten Ziel. Rückkehr vom Unterprogramm erfolgt durch Ausführung einer RETURN-Anweisung.

(2) Mehrfacher Unterprogrammabruf mit Bezug von einem Unterprogramm zu einem anderen Unterprogramm ist zulässig, soweit Speicherkapazität zur Verfügung steht.

(3) Das Unterprogramm darf keinen der folgenden Befehle enthalten:
CLEAR, LOAD

Beispiel

```
 10 PRINT "MAIN(10)"
 20 GOSUB 50
 30 PRINT "MAIN(20)
 40 END
 50 PRINT "SUB 100"
 60 GOSUB 90
 70 PRINT "SUB 100/RETURN"
 80 RETURN
 90 PRINT "SUB 200"
100 PRINT "SUB 200/RETURN"
110 RETURN
```

```
RUN
MAIN(10)
SUB 100
SUB 200
SUB 200/RETURN
SUB 100/RETURN
MAIN(20)
```

5-1-4-2. RETURN-Anweisung

```
RETURN
```

Funktion

Verursacht Rückkehr von einem Unterprogramm.

Erklärung

(1) Diese Anweisung verursacht Ruckkehr zu der Anweisung direkt nach der Anweisung, mit der das Unterprogramm abgerufen wurde.

Beispiel

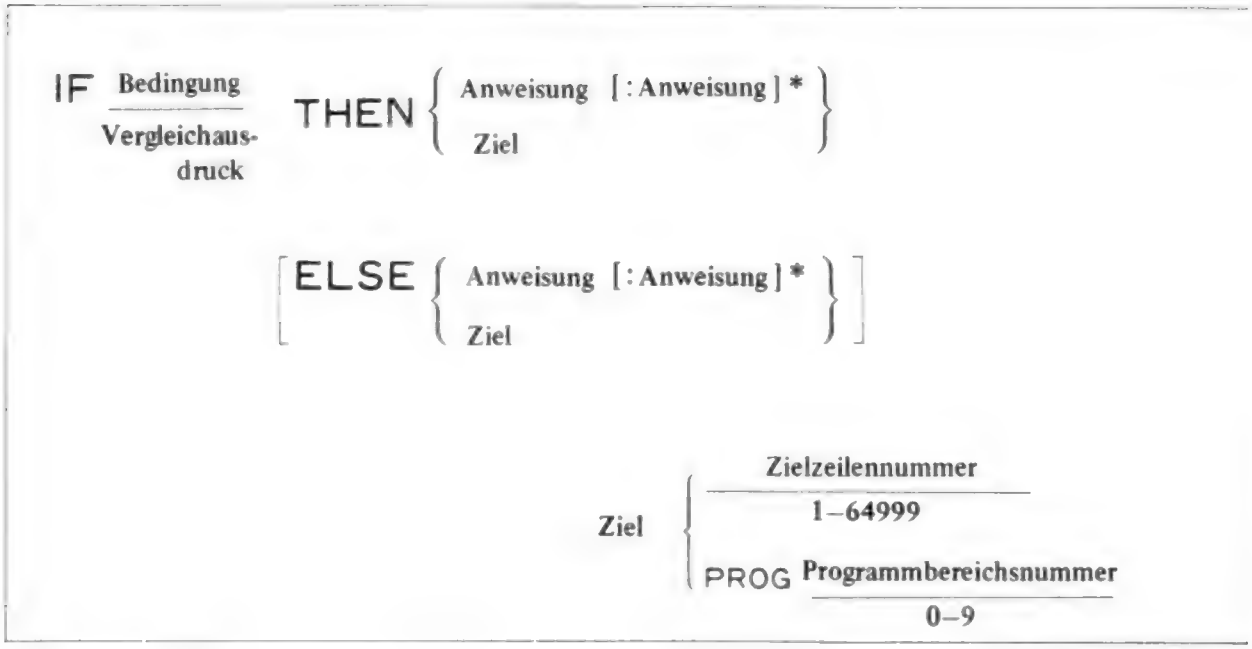
```
 10 PRINT "MAIN"
 20 GOSUB 140
 30 PRINT "MAINEND"
 40 END
140 PRINT "SUBROUTINE"
150 PRINT "RETURN"
160 RETURN
```

```
RUN
MAIN
SUBROUTINE
RETURN
MAINEND
```

Siehe in diesem Zusammenhang auch

GOSUB
ON-GOSUB

5-1-5. IF-THEN-ELSE-Anweisung



Funktion

Ausführung der Anweisung(en) nach THEN oder Sprung zu dem Ziel nach THEN oder GOTO, wenn die Bedingung erfüllt ist.
Ausführung der Anweisung(en) nach ELSE oder Sprung zu dem Ziel nach ELSE, wenn die Bedingung nicht erfüllt ist.

Parameter

(1)	Bedingung:	Vergleichsausdruck
(2)	Zielzeilennummer:	Eine Zeilennummer von 1 bis 64999.
(3)	Programmbereichsnummer:	Eine Ziffer von 0 bis 9.

Erklärung

(1) Durch diese Anweisung erfolgt Ausführung der Anweisung(en) nach THEN oder ein Sprung zum bezeichneten Ziel,wenn die Bedingung erfüllt ist.

(2) Wenn die Bedingung nicht erfüllt ist, so erfolgt durch diese Anweisung Ausführung der Anweisung(en) nach ELSE oder ein Sprung zum bezeichneten Ziel.
Wenn die ELSE-Klausel ausgelassen wird, so wird die nachste Zeile ausgeführt.

(3) Wenn der Wert des Vergleichsausdrucks 0 ist (wenn der absolute Wert kleiner als 1E-99 ist), so ist die Bedingung nicht erfüllt; ansonsten ist die Sprungbedingung erfüllt.

(4) Eine IF-Anweisung kann eine andere IF-Anweisung enthalten. In diesem Fall wird ein THEN mit dem nachsten ELSE assoziiert. Das gleiche gilt für den Zusammenhang zwischen GOTO und ELSE.

Beispiel

```
10 A=0
20 PRINT"START"
30 A=A+1
40 IF A MOD 2=0 THEN PRINT A,"EVEN" ELSE PRINT A,"ODD"
50 IF A<11 THEN 30
60 END
```

```
RUN
START
 1              ODD
 2              EVEN
 3              ODD
 4              EVEN
 5              ODD
 6              EVEN
 7              ODD
 8              EVEN
 9              ODD
 10             EVEN
 11             ODD
```

```
10 FOR I=0 TO 20
20 IF I MOD 2=0 THEN IF I MOD 3=0 THEN PRINT I
   ELSE IF I MOD 5=0 THEN PRINT I
30 NEXT
```

```
RUN
 0
 6
 10
 12
 18
 20
```

5-1-6. ON-GOTO-Anweisung

ON { Bedingung / Numerischer Ausdruck } **GOTO** { [(Ziel) [,[Ziel]]*, Ziel / Ziel }

Ziel : { Zielzeilennummer / Zeilennummer / PROG Programmbereichsnummer (n = 0 – 9) }

Funktion

Verursacht einen Sprung zum bezeichneten Ziel entsprechend der Bedingung.

Parameter

(1) Bedingung: Ein als ganze Zahl bewerteter numerischer Ausdruck.
(2) Zielzeilennummer: Eine Zeilennummer von 1 bis 64999.
(3) Programmbereichsnummer: Eine Ziffer von 0 bis 9.

Erklärung

(1) Diese Anweisung verursacht einen Sprung entsprechend dem ganzzahligen Teil des Wertes des numerischen Ausdrucks (Bedingung). Wenn der Wert "n" ist, so erfolgt ein Sprung zum n-ten Ziel in der Liste.

(2) Wenn der ganzzahlige Teil des Wertes des numerischen Ausdrucks weniger als 1 ist oder wenn kein entsprechendes Ziel vorhanden ist, so wird die nächste Anweisung ausgeführt. (In diesem Fall erfolgt kein Sprung.)

(3) Bis zu 99 Sprünge können bezeichnet werden.

Beispiel

```
10 FOR I=-5 TO 5
20 PRINT I,
30 ON I GOTO,,50,60,70
40 GOTO 80
50 PRINT "LINE 50",:GOTO 80
60 PRINT "LINE 60",:GOTO 80
70 PRINT "LINE 70",:GOTO 80
80 PRINT
90 NEXT I
100 END
```

```
RUN
-5
-4
-3
-2
-1
 0
 1
 2
 3              LINE 50
 4              LINE 60
 5              LINE 70
```

5-1-7. ON-GOSUB-Anweisung

```
ON  Bedingung  GOSUB { Ziel , [ Ziel ] [,[ Ziel ] ]*, Ziel }
    Numerischer        Ziel
    Ausdruck

                                  Zielnummer
                        Ziel {    1–64999
                                  PROG Programmbereichsnummer
                                       0–9
```

Funktion

Abruf des Unterprogramms am bezeichneten Ziel entsprechend der Bedingung.

Parameter

(1)	Bedingung:	Ein als ganze Zahl bewerteter numerischer Ausdruck.
(2)	Zielzeilennummer:	Eine Zeilennummer von 1 bis 64999.
(3)	Programmbereichsnummer:	Eine Ziffer von 0 bis 9.

Erklärung

(1) Diese Anweisung verursacht einen Unterprogrammsprung entsprechend dem ganzzahligen Teil des Wertes des numerischen Ausdrucks (Bedingung). Wenn der Wert "n" ist, so erfolgt ein Unterprogrammsprung zum n-ten Ziel in der Liste.

(2) Wenn der ganzzahlige Teil des Wertes des numerischen Ausdrucks weniger als 1 ist oder wenn kein entsprechendes Ziel vorhanden ist, so erfolgt kein Sprung und die nächste Anweisung wird ausgeführt.

(3) Bis zu 99 Ziele können bezeichnet werden.

Beispiel

```
 10 FOR I=1 TO 10
 20 PRINT I;TAB(5);
 30 ON I GOSUB 100,,200,,300,,,400
 40 PRINT
 50 NEXT I
 60 END
100 PRINT "SUB 100",:RETURN
200 PRINT "SUB 200",:RETURN
300 PRINT "SUB 300",:RETURN
400 PRINT "SUB 400",:RETURN
```

```
RUN
 1          SUB 100
 2
 3          SUB 200
 4
 5          SUB 300
 6
 7
 8          SUB 400
 9
 10
```

Siehe in diesem Zusammenhang auch

RETURN

5-1-8. FOR-NEXT-Anweisung

```
FOR Steuervariablenname = Ausgangswert TO Endwert [STEP Zuwachs ]
                          Numerischer      Numerischer
                            Ausdruck         Ausdruck
NEXT [ Steuervariablenname [, Steuervariablenname ]*]
```

Funktion

Wiederholte Ausführung der Anweisungen zwischen der FOR-Anweisung und der NEXT-Anweisung, während die Steuervariable mit dem bezeichneten Zuwachs vom Ausgangswert zum Endwert geandert wird. Wenn die Steuervariable den Endwert überschreitet, so wird die Wiederholung beendet.

Parameter

(1)	Steuervariablenname:	Ein einfacher Variablenname vom ganzzahligen Typ oder vom reellen Typ einfacher Genauigkeit.
(2)	Ausgangswert:	Ein numerischer Ausdruck.
(3)	Endwert:	Ein numerischer Ausdruck.
(4)	Zuwachs:	Ein numerischer Ausdruck. Der Vorgabewert ist 1.

Erklärung

(1) Durch diese Anweisung wird die Ausführung der Anweisungen zwischen der FOR-Anweisung und der NEXT-Anweisung wiederholt, während die Steuervariable um den bezeichneten Zuwachs vom Ausgangswert bis zum Endwert geändert wird. Wenn die Steuervariable den Endwert überschreitet, so wird die Wiederholung beendet.

(2) Wenn der Ausgangswert größer als der Endwert ist, so werden die Anweisungen zwischen FOR und NEXT überhaupt nicht ausgeführt, und die Anweisung nach der NEXT-Anweisung wird ausgeführt.

(3) Der Vorgabewert für den Zuwachs ist 1.

(4) Für jede FOR-Anweisung muß eine NEXT-Anweisung vorhanden sein. Die NEXT-Anweisung muß der zugehörigen FOR-Anweisung folgen.

(5) Eine FOR-NEXT-Schleife kann wie nachfolgend gezeigt in einer anderen FOR-NEXT-Schleife eingebettet sein.

```
10 FOR I=1 TO 12 STEP 3
20   FOR J=1 TO 4 STEP 0.5
30    PRINT I,J
40   NEXT J
50 NEXT I
60 END
```

(6) Die Anzahl der verschachtelten FOR-NEXT-Schleifen hängt von der Speicherkapazität ab.

(7) Wenn der Steuervariablenname in der NEXT-Anweisung selbstverständlich ist, so kann der Name ausgelassen werden.

(8) Fortlaufende NEXT-Anweisungen können durch Trennung ihrer Steuervariablennamen wie gezeigt durch Kommas (,) zu einer einzigen NEXT-Anweisung verringert werden.

```
10 FOR I=1 TO 12 STEP 3
20   FOR J=1 TO 4 STEP 0.5
30     PRINT I,J
40   NEXT J
50 NEXT I
60 END
```

⇨

```
10 FOR I=1 TO 12 STEP 3
20   FOR J=1 TO 4 STEP 0.
30     PRINT I,J
40 NEXT J,I
50 END
```

(9) Wenn der Wert der Steuervariablen den Endwert überschreitet, so wird die Schleife beendet.

(10) Es ist möglich, einen Sprung aus einer FOR-NEXT-Schleife durchzuführen. In diesem Fall wird der Steuervariablenwert bewahrt, so daß die Schleife durch eine GOTO-Anweisung usw. wieder aufgenommen werden kann.

(11) Wenn die Steuervariable der FOR-Anweisung mit der Steuervariablen der äußeren Schleife übereinstimmt, so werden beide Schleifen aufgegeben und es wird eine neue Schleife ausgeführt.
Normalerweise werden FOR-NEXT-Anweisungen nicht auf diese Weise verwendet. Bei Sprüngen aus einer Schleife durch eine IF-Anweisung, eine GOTO-Anweisung usw. sollte man deshalb Sorgfalt walten lassen.

```
10 FOR I=1 TO 10
20   FOR J=1 TO 10
30     FOR K=1 TO 10
40       IF K=5 THEN 80
50   NEXT K,J
60 NEXT I
70 END
80 A=3
90 FOR J=1 TO 5
100   PRINT J
110 NEXT J
120 GOTO 60
```

J-Schleife
I-Schleife
K-Scheleife

1) In I-, J- und K-Schleife

2) J- und K-Schleife werden abgebrochen, und es werden neue J-Schleifen gebildet.

5-2. Kommentare

5-2-1 REM-Anweisung

{ REM / ' } Kommentar
Zeichenkette

Funktion

Einschluß eines Kommentars im Programm. Es wird keine Operation bewirkt.

Parameter

(1) Kommentar — Eine Zeichenkette

Erklärung

(1) Diese Anweisung druckt einen Kommentar aus, aber sie führt keine Operation durch.

(2) Ein Komma (,) kann in der Anweisung verwendet werden, um anzuzeigen, daß die nachfolgende Anweisung ein Kommentar ist.

(3) Der REM-Anweisung kann keine weitere Anweisung in der gleichen Zeile folgen, da jegliche folgende Anweisung als Teil des Kommentars behandelt wird.

Beispiel

```
10 REM sum of 1-1000
20 L=1                  'initial set
30 S=0                  'same
40 FOR L=1 TO 1000      'while 1<=>1000 do
50 S=S+L                '
60 L=L+1                '
70 NEXT                 'loop here!!
80 PRINT S              'print out (sum of 1-1000)
90 END                  'end
```

5-3. Datenmanipulation

5-3-1. LET-Anweisung

[LET] { Numerischer Variablenname = Numerischer Ausdruck
Zeichenvariablenname = Zeichenausdruck }

Funktion

Zuordnung des Wertes des Ausdrucks auf der rechten Seite des Gleichheitszeichens (=) zu der Variablen auf der linken Seite des Gleichheitszeichens (=).

Erklärung

(1) Diese Anweisung ordnet der Variablen den Wert des Ausdrucks zu.

(2) Eine numerische Variable wird mit einem numerischen Ausdruck in Verbindung gebracht, und eine Zeichenvariable wird mit einem Zeichenausdruck in Verbindung gebracht. Wenn beide Seiten vom numerischen Typ sind, aber verschiedene Genauigkeit haben, so erfolgt automatisch Anpassung auf die gleiche Genauigkeit.

(3) LET kann ausgelassen werden.

Beispiel

```
10 LET X=12
20 Y=X*X+2*X-1        'LET Y=X*X+2*X-1
30 PRINT X;Y
40 END
```

```
RUN
 12       167
```

5-4. Von einem Programm gelesene Daten

5-4-1. DATA-Anweisung

DATA [Daten] [,[Daten]]*

Konstante Konstante

Funktion

Bezeichnung von Daten

Parameter

(1) Daten — Eine Zeichenkonstante oder eine numerische Konstante. Wenn eine Zeichenkonstante kein Komma (,) enthält, so können die Anführungszeichen (") auf beiden Seiten der Zeichenkonstante ausgelassen werden. Wenn dieser Parameter ausgelassen wird, so wird eine Zeichenkette mit der Länge 0 angenommen.

Erklärung

(1) Diese Anweisung wird verwendet, um Daten in einem durch eine READ-Anweisung zu lesenden Programm einzuschließen.

(2) Durch Trennen mittels Kommas können mehrere Datenpunkte geschrieben werden.

(3) Wenn keine "Datenparameter" voreingestellt sind, so wird eine Zeichenkette mit der Länge 0 angenommen, also

(4) Diese Anweisung führt keine Operation durch.

Beispiel

```
10 DATA 1
20 READ A,B,C
30 DATA 2,3,START,END
40 READ A$,B$
50 PRINT A,B,C
60 PRINT A$;"/";B$
70 END
```

```
RUN
 1   2   3
START/END
```

Siehe in diesem Zusammenhang auch

READ-Anweisung, RESTORE-Anweisung

5-4-2. READ-Anweisung

READ Variablenname [, Variablenname]*

Funktion

Lesen von Daten aus einer READ-Anweisung in eine festgelegte Variable (oder mehrere Variablen).

Parameter

(1) Variablenname

Erklärung

(1) Durch diese Anweisung werden Daten von der gegenwärtigen DATA-Anweisung in die bezeichneten Variablen gelesen.

(2) Der Variablentyp muß dem Typ der zugeordneten Daten entsprechen. Wenn beide numerische Typen, aber von unterschiedlicher Genauigkeit sind, so wird Typumwandlung automatisch durchgeführt.

(3) Datenpunkte in DATA-Anweisungen werden in steigender Reihenfolge der Zeilennummern gelesen. Datenpunkte in der gleichen DATA-Anweisung werden der Reihe nach vom Anfang her gelesen.

(4) Nach Lesen der festgelegten Anzahl der Datenpunkte werden die folgenden Datenpunkte durch die nächste READ-Anweisung gelesen.

Beispiel

```
10 DATA 1,2
20 READ A,B,C
30 READ D,E
40 DATA 3,4,5
45 PRINT A
46 PRINT B
47 PRINT C
48 PRINT D
49 PRINT E
50 END
```

```
RUN
 1
 2
 3
 4
 5
```

(5) Wenn die erste READ-Anweisung ausgeführt wird, so wird der erste Datenpunkt in dem Programmbereich, der die READ-Anweisung enthält, gelesen. Anschließend werden die nachfolgenden Datenpunkte in dem Programmbereich der Reihe nach gelesen.

(6) Durch Verwendung der RESTORE-Anweisung kann die DATA-Anweisung bezeichnet werden, aus der Daten zu lesen sind.

Siehe in diesem Zusammenhang auch

DATA-Anweisung, RESTORE-Anweisung

5-4-3. RESTORE-Anweisung

RESTORE [{ Zeilennummer / (Numerischer Ausdruck) }]

Funktion

Bezeichnung der Position der durch eine READ-Anweisung zu lesenden Daten.

Parameter

(1) Zeilennummer: Eine Zeilennummer von 1 bis 64999.

(2) Numerischer Ausdruck: Ein als ganze Zahl bewerteter numerischer Ausdruck. $1 \leqslant$ numerischer Ausdruck < 65000.

Erklärung

(1) Diese Anweisung bezeichnet die nachste durch eine READ-Anweisung zu lesende DATA-Anweisung.

(2) Wenn der Parameter ausgelassen wird, so liest die erste auszufuhrende READ-Anweisung von der ersten DATA-Anweisung in dem Programmbereich, der die READ-Anweisung enthält.

(3) Wenn eine Zeilennummer bezeichnet wird, so bezieht sich diese auf den Programmbereich, der die RESTORE-Anweisung enthalt. Nachfolgende READ-Anweisungen lesen die Daten in diesem Programmbereich der Reihe nach.

(4) Es ist moglich, eine Zeilennummer durch den Wert eines numerischen Ausdrucks zu bezeichnen, aber dieser Ausdruck wird durch RENUM nicht beeinflußt, so daß in diesem Fall Vorsicht für die Neunumerierung von Programmzeilen für DATA-Anweisungen erforderlich ist.

Beispiel

```
 10 DATA 1,2,3
 20 DATA 1,2
 30 READ A,B,C,D,E
 40 RESTORE 10
 50 READ F,G
 60 PRINT "A=";A,"B=";B
 70 PRINT "C=";C,"D=";D
 80 PRINT "E=";E,"F=";F
 90 PRINT "G=";G
100 END
```

```
RUN
A= 1          B= 2
C= 3          D= 1
E= 2          F= 1
G= 2
```

```
10 DATA DOG
20 DATA CAT
30 DATA COW
40 FOR I=3 TO 1 STEP -1
50 RESTORE (I*10)
60 READ A$
70 PRINT I,A$
80 NEXT I
90 END
```

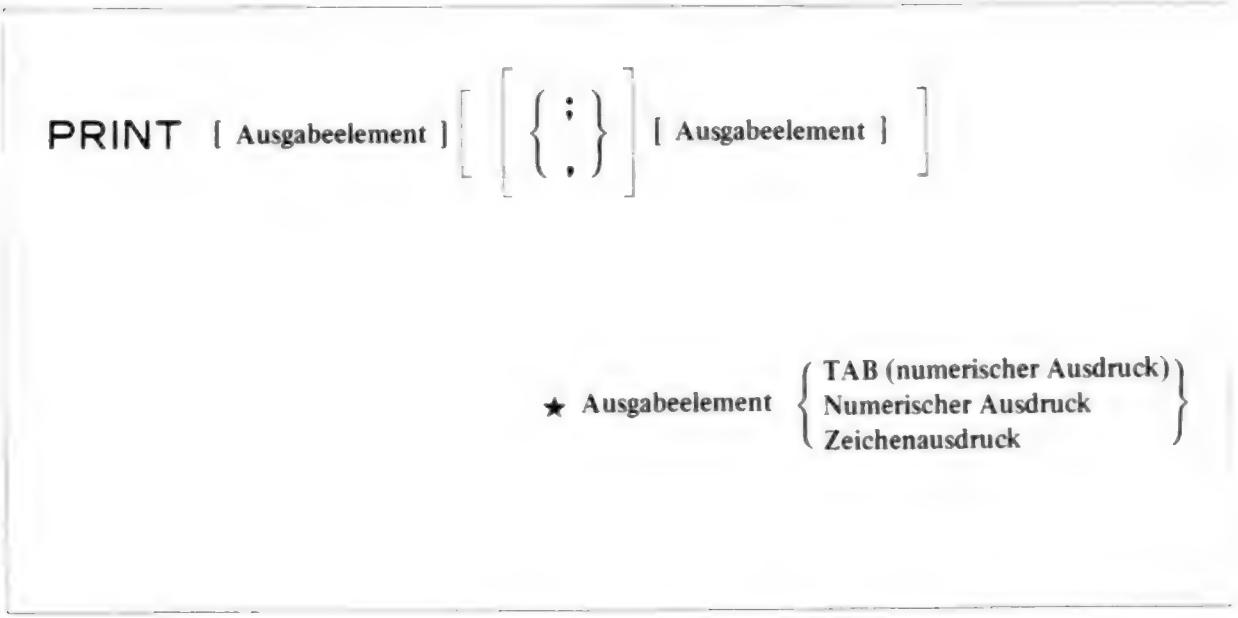
```
RUN
3             COW
2             CAT
1             DOG
```

Siehe in diesem Zusammenhang auch

DATA-Anweisung, READ-Anweisung

5-5. Anzeige

5-5-1. PRINT-Anweisung



Funktion

Anzeige von Zeichen auf der Flüssigkristallanzeige.

Parameter

(1) Ausgabeelement

Eine Ausgabesteuerfunktion oder ein numerischer Ausdruck oder ein Zeichenausdruck.

Erklärung

(1) Durch diese Anweisung wird das Ausgabeelement auf der Flüssigkristallanzeige angezeigt. Wenn das Ausgabeelement eine Ausgabesteuerfunktion ist, so wird die der Ausgabesteuerfunktion zugeordnete Operation durch PRINT ausgeführt. Wenn das Ausgabeelement ein numerischer Ausdruck oder ein Zeichenausdruck ist, so wird der Wert des Ausdrucks durch PRINT angezeigt.

(2) Der Wert eines numerischen Ausdrucks wird in Dezimalnotierung angezeigt. Die Anzahl der angezeigten Stellen ist kleiner als die Anzahl der inneren Stellen. Die Stellen werden mit einer nachfolgenden Leerstelle angezeigt.

Es gibt drei Arten von Anzeigeformaten: Ganzzahlige Anzeige, Festkommaanzeige und Gleitkommaanzeige. Das Anzeigeformat wird automatisch entsprechend dem anzuzeigenden Typ und Wert gewählt.

1) Reeller Typ einfacher Genauigkeit

Das durch Runden der siebten Stelle der Mantisse erhaltene Ergebnis wird angezeigt.

a) Eine ganze Zahl kleiner als 1E6: . . . . . Ganzzahlige Anzeige.
b) Sechs oder weniger Kommastellen: . . . . . Festkommaanzeige.
c) Andere als a) und b): . . . . . Gleitkommaanzeige.

2) Reeller Typ doppelter Genauigkeit

Das durch Runden der siebzehnten Stelle der Mantisse erhaltene Ergebnis wird angezeigt.

a) Eine ganze Zahl kleiner als 1E16: . . . . . Ganzzahlige Anzeige.

b) Sechzehn oder weniger Kommastellen: . . . . . Festkommaanzeige.

c) Andere als a) und b): . . . . . Gleitkommaanzeige.

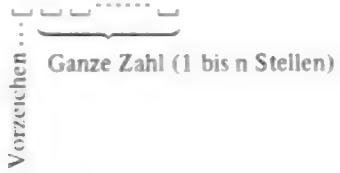
Die einzelnen Anzeigeformate sind nachfolgend gezeigt. Der Wert von "n" in der Abbildung ändert sich wie folgt entsprechend dem Datentyp.

n = 6 für den reellen Typ einfacher Genauigkeit

n = 16 für den reellen Typ doppelter Genauigkeit

Das Vorzeichen ist ein Minuszeichen für negative Daten und eine Leerstelle für positive Daten.

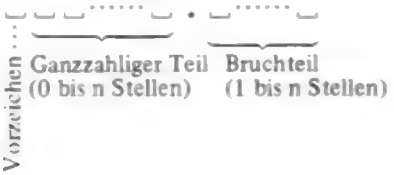
1) Ganzzahlige Anzeige



Beispiel

−1245

2) Festkommaanzeige



Beispiel

␣1.12571

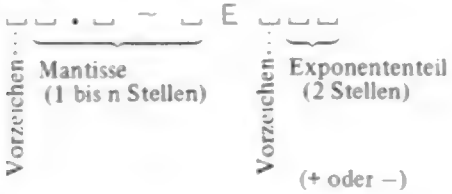
−.123456

3) Gleitkommaanzeige

Beispiel

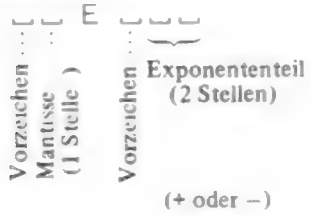
−1.23E+03

Mantissa mit mehr als 1 Stelle.



Einstellige Mantisse:

␣2E−15



(Dies bedeutet, daß kein Dezimalpunkt für eine einstellige Mantisse angezeigt wird.)

Die Mantisse beginnt mit einer Ziffer, die nicht Null ist.

(3) Der Wert eines Zeichenausdrucks wird direkt angezeigt.

(4) Das Ausgabeelement wird rechts von der gegenwärtigen Position des Positionsanzeigers angezeigt. Wenn die Anzeige den rechten Rand des Bildschirms erreicht, so wird Zeilenvorschub ausgeführt, und die Anzeige wird auf der nächsten Zeile fortgesetzt. Wenn ein Zeilenvorschub auf der untersten Zeile durchgeführt wird, so wird der Bildschirm um eine Zeile nach oben gerollt, um Fortsetzung der Anzeige zu ermöglichen.

(5) Wenn Ausgabeelemente durch ein Komma (,), getrennt sind, so wird zwischen den einzelnen Ausgabeelementen jeweils eine Zonentabulierung durchgeführt. Jeder Abschnitt von 10 Zeichen einer Zeile auf der Flüssigkristallanzeige wird als eine Zone bezeichnet. Durch Zonentabulierung wird der Positionsanzeiger von der gegenwärtigen Position zum Anfang der nächsten Zone bewegt. Auf diese Weise wird ein auf ein Komma (,) folgendes Ausgabeelement immer am Anfang einer Zone angezeigt. Dies ermöglicht geordnete Anzeige der Ausgabeelemente.

Wenn die erste Zeichenposition als die Position 0 bezeichnet wird, so beginnen die Zonen mit den Zeichenpositionen 0 und 10.

```
10 PRINT "1st","2nd","3rd",
20 FOR I=4 TO 6
30 PRINT MID$(STR$(I)+"th",2),
40 NEXT I
50 PRINT
60 END
```

```
RUN
1st         2nd
3rd         4th
5th         6th
```

(6) Wenn Ausgabeelemente durch ein Semikolon (;) getrennt sind, so werden sie ohne Tabulierung direkt aufeinander folgend augezeigt.

```
10 FOR I=1 TO 10
20 PRINT "(";I;")";
30 NEXT
40 PRINT
50 END
```

```
RUN
( 1)( 2)( 3)( 4)
( 5)( 6)( 7)( 8)
( 9)( 10)
```

(7) Ausgabeelemente können durch eine Leerstelle getrennt werden. Wenn die individuellen Ausgabeelemente unzweideutig erkannt werden können, so kann auch das Abgrenzungszeichen ausgelassen werden. In jedem Fall ist der Betrieb der gleiche wie bei Verwendung eines Semikolons (;) als Abgrenzungszeichen.

Beispiel

```
PRINT   A␣B

PRINT   SIN(30) COS(30)
```

(8) Durch ein Semikolon (;) am Ende der Anweisung wird der Positionsanzeiger an der letzten Anzeigeposition angehalten.

(9) Durch ein Komma (,) am Ende der Anweisung wird eine Zonentabulierung ausgeführt.

(10) Wenn am Ende der Anweisung weder ein Semikolon noch ein Komma verwendet wird, so wird nur Zeilenvorschub durchgeführt.

```
10 PRINT "1st PRINT",
20 PRINT "2nd PRINT"
30 PRINT "3rd PRINT";
40 PRINT "4th PRINT"
50 END
```

```
RUN
1st print 2nd print
3rd print4th print
```

TAB (Tabulatorspezifikation)

Numerischer Ausdruck

Funktion

Der Positionsanzeiger wird horizontal zu der festgelegten Position auf der Flüssigkristallanzeige bewegt.

Parameter

(1) Tabulatorspezifikation — Ein als ganze Zahl bewerteter numerischer Ausdruck. Der Bereich des numerischen Ausdrucks ist wie folgt: $0 \leqslant$ Tabulatorposition < 256

Erklärung

(1) Diese Funktion wird in der PRINT-Anweisung verwendet, um eine Anzeigeposition in einer Zeile auf der Flussigkristallanzeige festzulegen. Die Positionen zwischen der gegenwartigen Anzeigeposition und der festgelegten Position werden mit Leerstellen gefüllt.

(2) Die Anzeigeposition wird wie folgt gegeben:

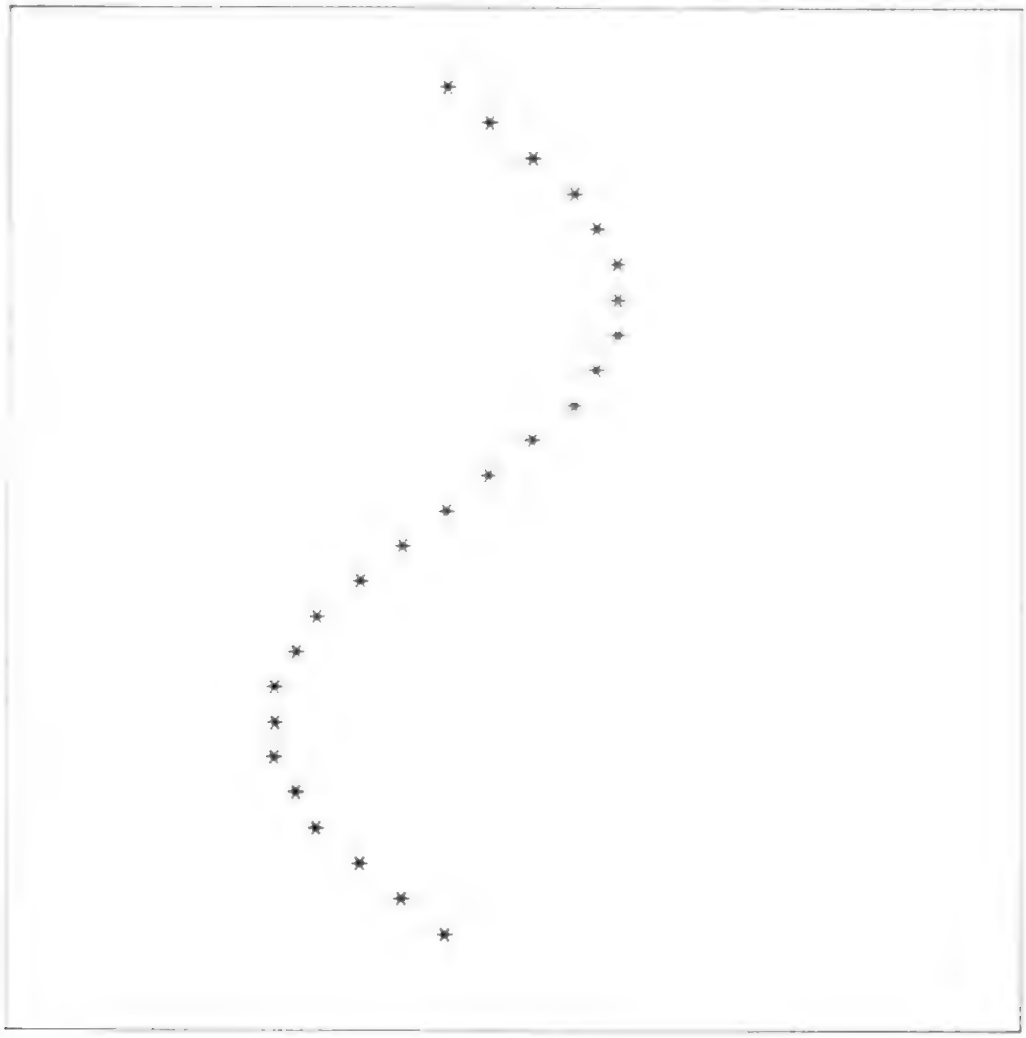
1) Die Positionen sind von links nach rechts numeriert, beginnend mit 0.

2) Wenn eine Position links von der gegenwärtigen Anzeigeposition festgelegt wird, so wird ein Zeilenvorschub durchgeführt, und die Position bezieht sich auf die nächste Zeile.

3) Wenn eine Position über das Ende der gegenwärtigen Zeile hinaus festgelegt wird, so wird eine durch Zählen vom Anfang der gegenwärtigen Zeile bestimmte Position festgelegt.

Beispiel

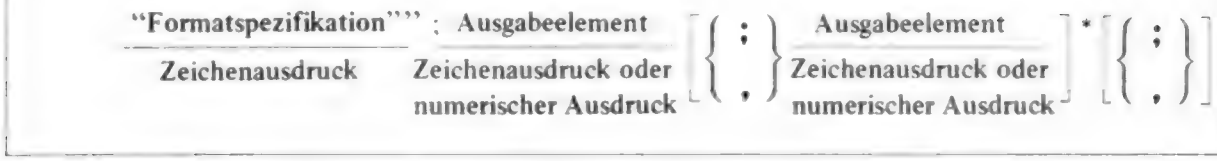
```
10 CLS:ANGLE 0
20 FOR X=0 TO 360 STEP 15
30 B$="*"
40 Y=INT(SIN(X)*8+10.5):YO=10
50 IF Y=YO THEN PRINT TAB(10);"*":GOTO 70
60 PRINT TAB(Y);B$
70 NEXT X
80 END
```



5-5-3. PRINT-USING-Anweisung

PRINT USING

$$\underbrace{\text{"Formatspezifikation"}}_{\text{Zeichenausdruck}}\text{"}\ ;\ \underbrace{\text{Ausgabeelement}}_{\text{Zeichenausdruck oder numerischer Ausdruck}}\left[\left\{\begin{matrix};\\,\end{matrix}\right\}\underbrace{\text{Ausgabeelement}}_{\text{Zeichenausdruck oder numerischer Ausdruck}}\right]^{*}\left[\left\{\begin{matrix};\\,\end{matrix}\right\}\right]$$



Funktion

Anzeige der Ausgabeelemente entsprechend der Formatspezifikation.

Parameter

(1) Formatspezifikation — Eine aus einem oder mehreren Zeichen bestehende Zeichenkette.

(2) Ausgabeelement — Ein Zeichenausdruck oder ein numerischer Ausdruck.

Erklärung

(1) Durch diese Anweisung werden Ausgabeelemente entsprechend der Formatspezifikation angezeigt. Das Format wird durch eine Kombination der folgenden Zeichen ausgedrückt.

1) Formate für Zeichenausgabeelemente

! . . . . . Anzeige nur des ersten Zeichens des Ausgabeelements.

&_____& . . Anzeige einer durch die beiden &-Symbole und die Anzahl der Leerstellen bestimmten Anzahl von Zeichen. Wenn das Ausgabeelement länger als die angezeigte Länge ist, so wird nur die bezeichnete Anzahl von Zeichen angezeigt. Wenn das Ausgabeelement kürzer als die bezeichnete Länge ist, so wird es links ausgerichtet angezeigt, und die verbleibenden Positionen auf der rechten Seite werden durch Leerstellen aufgefüllt.

Beliebige Anzahl von Leerstellen

@ . . . . . Das Ausgabeelement verbleibt unverändert.

2) Formate für numerische Ausgabeelemente

. . . . . Bezeichnung der Anzahl der anzuzeigenden Stellen. Der numerische Wert wird rechtsbündig ausgerichtet angezeigt.

. . . . . Bezeichnung der Position der Kommastelle. Wenn die Anzahl der #-Symbole nach der Kommastelle 0 ist, so wird 0 angezeigt.

+	Bei Verwendung am Ende der Formatspezifikation wird ein nachfolgendes Zeichen angezeigt. Für einen negativen Wert wird ein Minuszeichen (–) angezeigt, und für einen positiven Wert wird ein Pluszeichen (+) angezeigt. In einer Formatspezifikation kann nur ein Pluszeichen verwendet werden. Bei Verwendung am Anfang der Formatspezifikation wird entweder + oder – als Vorzeichen angezeigt.
–	Verwendung am Ende der Formatspezifikation. In dieser Position wird ein nachfolgendes Zeichen angezeigt. Für einen negativen Wert wird ein Minuszeichen (–) angezeigt, während für einen positiven Wert eine Leerstelle angezeigt wird. In einer Formatspezifikation kann nur ein Pluszeichen verwendet werden.
**	Verwendung am Anfang der Formatspezifikation. Die Zeichenpositionen vor den angezeigten Stellen werden durch Sterne (*) gefüllt.
$$	Verwendung am Anfang der Formatspezifikation. Direkt vor dem numerischen Wert wird ein $-Symbol angezeigt.
**$	Verwendung am Anfang der Formatspezifikation. Die Zeichenpositionen vor den angezeigten Stellen werden durch Sterne (*) gefüllt und direkt vor dem numerischen Wert wird ein $-Symbol angezeigt.
,	Positionierung in einer Anordnung von #-Symbolen. In der entsprechenden Position wird ein Komma angezeigt, wenn Stellen links davon vorhanden sind.
^^^^	Verwendung am Ende der Formatspezifikation. Anzeige des Exponenten des numerischen Wertes.

(2) Wenn ein numerischer Wert die Anzahl der durch das Format festgelegte Anzahl von Zeichen literal angezeigt.

(3) Eine Zeichenformatspezifikation muß einem Zeichenausgabeelement entsprechen, und eine numerische Formatspezifikation muß einem numerischen Ausgabeelement entsprechen.

(4) Wenn ein numerischer Wert die Anzahl der durch das Format festgelegte Anzahl von Stellen überschreitet, so wird er auf die festgelegte Anzahl von Stellen gerundet und angezeigt. Wenn der Wert zu klein ist, um in dem festgelegten Format ausgedrückt zu werden, so wird er mit einem %-Zeichen davor ohne Berücksichtigung des festgelegten Formats angezeigt.

(5) Die Formatspezifikation kann mehr als ein Format enthalten. In diesem Fall sollten die Formate durch ein in den Formaten nicht verwendetes Zeichen getrennt werden.

(6) Wenn die Anzahl der Ausgabeelemente größer als die Anzahl der Formate ist, so werden die Formate wiederholt der Reihe nach von Anfang an verwendet.

(7) Formate, für die kein Ausgabeelement vorhanden ist, werden nicht angezeigt.

(8) Wenn die Anweisung weder mit einem Semikolon (;) noch mit einem Komma (,) endet, so wird nach Beendigung der Anzeige ein Zeilenvorschub durchgeführt.

(9) Wenn "^^^^" in einer Formatspezifikation verwendet wird, so wird ein vorangehendes "*" als "##" angesehen.

Beispiel

```
 10 A$="CASIO":R=123.456
 20 PRINT USING "!";A$
 30 PRINT USING "&  &";A$
 40 PRINT USING "THE HANDHELD COMPUTER @";A$
 50 PRINT USING " ####.####";R,-R
 60 PRINT USING " +###.####";R,-R
 70 PRINT USING " ####.#+  ";R,-R
 80 PRINT USING " ####.#-  ";R,-R
 90 PRINT USING " **##.#-  ";R,-R
100 PRINT USING " $$##.#-  ";R,-R
110 PRINT USING " **$#.#+  ";R,-R
120 PRINT USING " ##,#.##  ";R,-R
130 PRINT USING " +#,#.##  ";R,-R
140 PRINT USING " ###.#####^^^^  ";R,-R
150 PRINT USING " ANS=###.###^^^^";R,-R
```

```
C
CASI
THE HANDHELD COMPUTER CASIO
  123.4560 -123.4560
 +123.4560 -123.4560
  123.5+     123.5-
  123.5      123.5-
 *123.5     *123.5-
 $123.5     $123.5-
 $123.5+    $123.5-
  123.46    -123.46
 +123.46    -123.46
  12.34560E+01    -12.34560E+01
 ANS= 12.346E+01 ANS=-12.346E+01
```

Siehe in diesem Zusammenhang auch

PRINT, LPRINT, LPRINT-USING

5-5-4. LOCATE-Anweisung

LOCATE X-Koordinate, Y-Koordinate
Numerischer Ausdruck

Funktion

Positionieren des Positionsanzeigers auf der Flüssigkristallanzeige.

Parameter

(1) X-Koordinate — Ein als ganze Zahl bewerteter numerischer Ausdruck. Der Bereich des numerischen Ausdrucks ist wie folgt:
$0 \leqslant$ X-Koordinate < 256.

(2) Y-Koordinate — Ein als ganze Zahl bewerteter numerischer Ausdruck. Der Bereich des numerischen Ausdrucks ist wie folgt:
$0 \leqslant$ Y-Koordinate < 256.

Erklärung

(1) Diese Anweisung legt die Position des Positionsanzeigers auf der Flüssigkristallanzeige fest.

(2) Die X-Koordinate wird von links nach rechts und die Y-Koordinate wird von oben nach unten gegeben, wobei der Ursprung(0, 0) an der linken oberen Ecke des Bildschirms ist.

(3) Wenn die bezeichnete X-Koordinate den rechten Rand des Bildschirms überschreitet, so wird der rechte Rand als die bezeichnete X-Koordinate angesehen.

(4) Wenn die Y-Koordinate den unteren Rand des Bildschirms überschreitet, so wird der untere Rand als die bezeichnete Y-Koordinate angesehen.

Beispiel

```
10 CLS
20 FOR I=0 TO 100
30 W=RND(-1)*20:L=RND(-1)*8:LOCATE W,L
40 PRINT "*";
50 NEXT
```

5-5-5. CLS-Anweisung

```
CLS
```

Funktion

Löschen der Flüssigkristallanzeige.

Erklärung

(1) Diese Anweisung löscht die Flüssigkristallanzeige und bewegt den Positionsanzeiger zur linken oberen Ecke des Bildschirms (0, 0).

Beispiel

```
10 PRINT "BEFORE CLEAR"
20 CLS
30 END
```

5-6. Tastenfeldeingabe

5-6-1. INPUT-Anweisung

```
INPUT [ Fragestellung / Ein mit einer Zeichenkonstanten beginnender Zeichenausdruck { , / ; } ] Variablenname [ , Variablenname ]*
```

Funktion

Anforderung von Dateneingabe von der Tastatur.

Parameter

(1) Fragestellung — Ein mit einer Zeichenkonstanten beginnender Zeichenausdruck.

(2) Variablenname — Ein numerischer Variablenname oder ein Zeichenvariablenname.

Erklärung

(1) Diese Anweisung fordert die Bedienung zur Eingabe festgelegter Variablen vom Tastenfeld her auf.

(2) Wenn eine Fragestellung festgelegt wird, so wird sie angezeigt. Wenn ein Semikolon(;) einer Fragestellung folgt, so wird ein Fragezeichen (?) nach der Fragestellung angezeigt.

(3) Wenn keine Fragestellung festgelegt wird, so wird ein Fragezeichen (?) angezeigt.

(4) Drücken Sie nach der Eingabe von Daten die Taste **RETURN**. Wenn mehrere Datenpunkte einzugeben sind, so sollten diese durch Kommas getrennt werden.

(5) Wenn mehr Datenpunkte als angefordert eingegeben werden, so werden die überflüssigen Datenpunkte nicht beachtet.

(6) Wenn weniger Datenpunkte als angefordert eingegeben werden, so wird ein Fragezeichen (?) angezeigt, um die Bedienung zur Eingabe zusätzlicher Daten aufzufordern.

(7) Eingabe von Zeichendaten für eine numerische Variable verursacht einen TM-Fehler. Die Bedienung wird zu erneuter Eingabe der Daten aufgefordert.

(8) Das Format der Eingabedaten ist das gleiche wie für Konstanten in Programmen. Beachten Sie jedoch bitte, daß die Anführungszeichen (") auf beiden Seiten der Zeichendaten ausgelassen werden können, wenn die Zeichendaten kein Komma enthalten. Durch Druck der Taste **RETURN** ohne Eingabe von Daten wird deshalb eine Zeichenkette mit der Länge 0 eingegeben (d.h. gleichwertig mit " ".)

(9) Bei der Eingabe von numerischen Daten für eine numerische Variable wird Typumwandlung automatisch durchgeführt, wenn die Datentypen nicht übereinstimmen.

(10) Nur anzeigbare Zeichen können als Zeichendaten eingegeben werden, d.h. die internen Kode 20 bis 7E, 80 bis 9F und E0 bis FE.

(11) Die Daten werden über einen Tastenpuffer eingegeben.

Beispiel

```
10 INPUT A$
20 INPUT "A$=";A$
30 INPUT "A$=",A$
40 INPUT "A,B,C=";A,B,C
50 END
```

```
RUN
?SAMPLE
A$=?SAMPLE
A$=SAMPLE
A,B,C=?1,2,3
```

INKEY $

Funktion

Abtasten des Tastenfelds für Dateneingabe.

Erklärung

(1) Dies ist eine Zeichenfunktion,. die das erste Zeichen vom Tastenfeld her erhält. Wenn der Tastenpuffer leer ist (d.h. wenn keine Taste gedruckt ist), so wird eine Zeichenkette mit der Länge 0 erhalten.

(2) Alle Tasten können eingegeben werden, ausgenommen die Tasten **BREAK** und **STOP/CONT**. Beziehen Sie sich auf die Tasten-Kode-Beziehungstabelle am Ende dieses Handbuchs.

(3) Während der Ausführung dieser Funktion blinkt der Positionsanzeiger nicht. Die Eingabedaten werden nicht angezeigt. Beachten Sie bitte, daß selbst bei Eingabe eines Steuerkodes (00 bis 1F) die entsprechende Operation nicht ausgeführt wird.

Beispiel

```
 10 PRINT "SECRET KEY=";
 20 B$=""
 30 IF LEN(B$)=>8 THEN 100
 40 A$=INKEY$
 50 IF A$<>"" THEN 80
 60 A$=INKEY$
 70 GOTO 50
 80 B$=B$+A$
 90 GOTO 30
100 PRINT B$
110 END
```

```
RUN
SECRET KEY=WHAT KEY
```

5-7. Variablengruppen

5-7-1. OPTION-BASE-Anweisung

OPTION BASE Mindestwert / Numerischer Ausdruck

Funktion

Festlegung des Mindestwertes einer Variablengruppentiefstellung als 0 oder 1.

Parameter

(1) Mindestwert — Ein als ganze Zahl bewerteter numerischer Ausdruck. Der numerische Ausdruck muß 0 der 1 entsprechen.

Erklärung

(1) Diese Anweisung legt den Mindestwert von nachfolgend erklärten Variablengruppentiefstellungen als 0 oder 1 fest.

(2) Der Mindestwert vorhandener Variablengruppentiefstellungen verbleibt unverändert.

(3) Direkt nach Einschalten der Stromversorgung, RUN, RESET oder NEW ALL wird der Mindestwert von Variablengruppentiefstellungen als 0 angenommen.

Beispiel

```
10 DIM A(10)          'A(0)..A(10)
20 OPTION BASE 1
30 DIM B(10)          'B(1)..B(10)
40 FOR I=0 TO 10
50 A(I)=I
60 NEXT I
70 FOR I=1 TO 10
80 B(I)=I
90 NEXT I
100 END
```

Siehe in diesem Zusammenhang auch

DIM-Anweisung

5-7-2. DIM-Anweisung

DIM Variablengruppenname (Maximalwert der Tiefstellung [, Maximalwert der Tiefstellung] *)
Numerischer Ausdruck — Numerischer Ausdruck

[, Variablengruppenname (Maximalwert der Tiefstellung [, Maximalwert der Tiefstellung])] *
Numerischer Ausdruck — Numerischer Ausdruck

Funktion

Erklärung einer Variablengruppe.

Parameter

(1) Variablengruppenname

(2) Maximalwert der Tiefstellung
Ein als ganze Zahl bewerteter numerischer Ausdruck.
Der Bereich des numerischen Ausdrucks ist wie folgt:
$0 \leqslant$ numerischer Ausdruck für OPTION BASE = 0.
$1 \leqslant$ numerischer Ausdruck für OPTION BASE = 1.

Erklärung

(1) Diese Anweisung erklärt (reserviert Speicher für) eine Variablengruppe. Die Anzahl der für Tiefstellungen gegebenen Maximalwerte bestimmt die Dimension der Variablengruppe, und die Maximalwerte der Tiefstellungen bestimmen die Größe der Variablengruppe (d. h. der Speicheranforderungen).

(2) Der Mindestwert der Variablengruppentiefstellungen ist ein durch OPTION BASE bestimmter Wert. Wenn OPTION BASE nicht festgelegt ist, so sind die Mindestwerte 0.

(3) Wenn eine vorhandene Variablengruppe neu festgelegt wird, so sollten die folgenden Bedingungen erfüllt sein:

1) Die Dimension der neu erklärten Variablengruppe muß mit der Dimension der vorherigen Variablengruppe übereinstimmen.

2) Die Anzahl der Elemente der neu erklarten Variablengruppe muß mit der Anzahl der Elemente der vorherigen Variablengruppe übereinstimmen.

3) Der Mindestwert der Tiefstellung der neu erklärten Variablengruppe muß mit dem Mindestwert der Tiefstellung der vorherigen Variablengruppe übereinstimmen.

(4) Jedem Element einer neu erklärten Variablengruppe wird ein Anfangswert zugeordnet. Der Anfangswert ist eine Zeichenkette mit der Länge 0 für Zeichenvariablengruppen und die Zahl 0 für numerische Variablengruppen.

(5) Für Variablengruppen sind bis zu drei Dimensionen zulässig.

Beispiel

```
10 CLEAR
20 DIM A!(500)
30 FOR I=2 TO INT(SQR(500)+1)
40 IF A!(I)=-1 THEN 90
50 PRINT I,
60 FOR J=I+I TO 500 STEP I
70 A!(J)=-1
80 NEXT J
90 NEXT I
100 FOR I=I TO 500
110 IFF A!(I)=0 THEN PRINT I,
120 NEXT I
130 PRINT
140 END
```

```
RUN
 2            3
 5            7
 11           13
 17           19
 23
```

Siehe in diesem Zusammenhang auch

OPTION BASE

5-8. Abruf der Maschinensprache

5-8-1. CALL-Anweisung

CALL Adresse / Numerischer Ausdruck { [, [Argument]]*, Argument / [, Argument] }

Funktion

Abruf eines Maschinensprachenprogramms.

Parameter

(1) Adresse

Das Argument A (das erste Argument) ist ein als ganze Zahl bewerteter numerischer Ausdruck im Bereich von 0 bis 255.

(2) Argument A, HL, DE und BC

Die Adresse und die zu HL, DE und BC gegebenen Werte mussen als ganze Zahlen bewertete numerische Ausdrücke im Bereich von 0 bis 65535 sein.

Erklärung

(1) Diese Anweisung ruft ein Maschinensprachenprogramm ab. Verwenden Sie die RET-Anweisung (C9) der Assemblersprache, um von diesem Programm zurückzukehren.

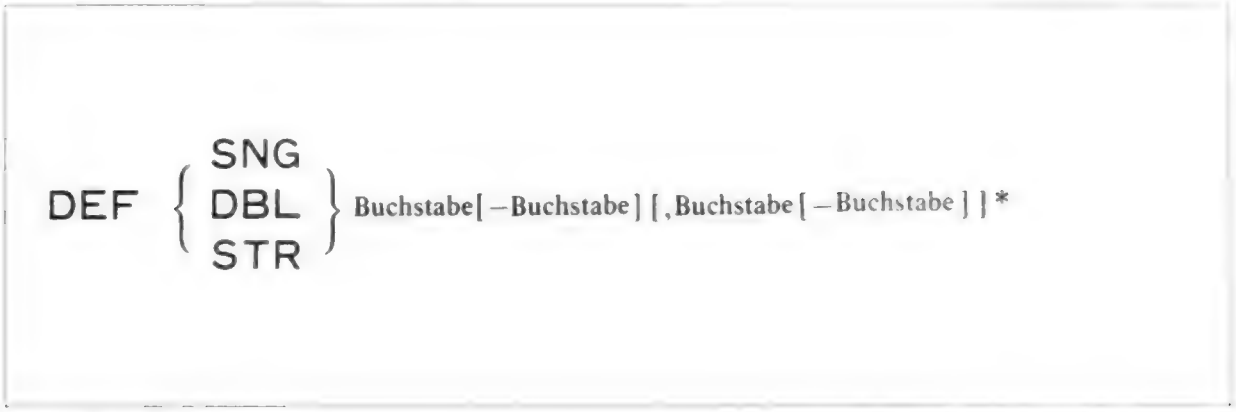
(2) Es ist möglich, Argumente über Register zum Maschinensprachenprogramm zu leiten. Argumente können den Registern A, HL, DE und BC in dieser Reihenfolge zugeordnet werden. Die Anweisung darf nicht mit einem Komma (,) enden.

Beispiel

```
10 FOR I=0 TO 128
20 CALL 30602,I .MOD 64
30 FOR J=1 TO 1:NEXT J
40 NEXT I
50 END
```

5-9. Typerklärungen

5-9-1. DEF SNG/DBL/STR-Anweisung



Funktion

Definition des Datentyps für Variablen und durch den Benutzer definierte Funktionen.

Parameter

(1) **ANG:** Definition des reellen Typs einfacher Genauigkeit.
DBL: Definition des reellen Typs doppelter Genauigkeit.
STR: Definition des Zeichentyps.

(2) Buchstabe

Ein Groß- oder Kleinbuchstabe von A bis Z.

Erklärung

(1) Definition des Datentyps für Variablen und durch den Benutzer definierte Funktionen.

(2) "Buchstabe" oder "Buchstabe-Buchstabe" reprasentiert den Umfang der Definition wie folgt:

1) Buchstabe

Alle mit diesem Buchstaben beginnenden Variablen und durch den Benutzer definierten Funktionen sind vom bezeichneten Datentyp.

2) Buchstabe-Buchstabe

Alle mit den in "Buchstabe-Buchstabe" bezeichneten Buchstaben beginnenden Variablen und durch den Benutzer definierten Funktionen sind vom bezeichneten Datentyp.

(3) Alle Variablen werden durch Einschalten der Stromversorgung, RUN, NEW ALL bzw. RESET auf den Typ mit einfacher Genauigkeit initialisiert.

Beispiel

```
 10 DEFDBL A-Z
 20 DEFSNG I
 30 PRINT "*** ROOT(X) ***"
 40 PRINT
 50 INPUT "VALUE=",X
 60 IF X=<0 THEN END
 70 MIN=0:MAX=X:IF MIN>MAX THEN WORK=MIN:MIN=MAX:MAX=WORK
 80 FOR I=1 TO 1000
 90 MEAN=(MIN+MAX)/2
100 XX=MEAN*MEAN
110 IF ABS((XX-X)/X)<1E-22 THEN 160
120 IF XX>X THEN MAX=MEAN ELSE MIN=MEAN
130 NEXT I
140 PRINT "NOT FOUND"
150 END
160 PRINT "ROOT=";MEAN
170 GOTO 50
```

CSNG (Argument)

Numerischer Ausdruck

CDBL (Argument)

Numerischer Ausdruck

Funktion

Umwandlung des Wertes des Arguments in den durch die Funktion bestimmten Datentyp.

Parameter

(1) Argument: Ein numerischer Ausdruck.

Erklärung

(1) Der Wert des Arguments wird in den durch die Funktion bestimmten Typ umgewandelt. Die Regeln für Typumwandlung sind die gleichen wie für automatische Umwandlung, d.h.:

CSNG: Umwandlung in reellen Typ einfacher Genauigkeit.
CDBL: Umwandlung in reellen Typ doppelter Genauigkeit.

Beispiel

```
10 FOR I=1 TO 5
20 PRINT SQR(CDBL(I))
30 NEXT I
40 END
```

```
RUN
 1
 1.414213562373095
 1.732050807568877
 2
 2.23606797749979
```

5-10. Mathematische Funktionen

5-10-1. Trigonometrische Funktionen

5-10-1-1. ANGLE-Anweisung

ANGLE Winkelbezeichnung

Numerischer Ausdruck

Funktion

Bezeichnung der Winkelbetriebsart für trigonometrische Funktionen.

Parameter

(1) Winkelbezeichnung
Ein numerischer Ausdruck. Nur der ganzzahlige Teil ist von Bedeutung, und er muß 0, 1 oder 2 sein.

Erklärung

(1) 0: DEG (Grad)
1: RAD (Bogenmaß)
2: GRAD (Neugrad)

(2) Der Zusammenhang zwischen den Winkelbetriebsarten ist wie folgt:

Betriebsart \ Winkel	DEG	RAD	GRAD
1DEG =	1	$\frac{\pi}{180}$	$\frac{100}{90}$
1RAD =	$\frac{180}{\pi}$	1	$\frac{200}{\pi}$
1GRAD =	$\frac{90}{100}$	$\frac{\pi}{200}$	1

$180^\circ = \pi$ rad $= 200$grad

(3) Im Ausgangszustand und nach Ausführung der Kommandos RESET und NEW ALL ist ANGLE auf 0 eingestellt.

Beispiel

```
10 ANGLE 0
20 PRINT SIN (30);
30 ANGLE 1
40 PRINT SIN (3.14159/6);
50 ANGLE 2
60 PRINT SIN (100/3)
```

```
RUN
 0.5  0.5  0.5
```

5-10-1-2. SIN/COS/TAN-Funktion

SIN (Winkel)
Numerischer Ausdruck

COS (Winkel)
Numerischer Ausdruck

TAN (Winkel)
Numerischer Ausdruck

Funktion

Angabe des trigonometrischen Funktionswertes für einen gegebenen Winkel.

Parameter

(1) Winkel
Ein numerischer Ausdruck.
$-1440^\circ <$ Winkel $< 1440^\circ$ (Grad)
$-8\pi <$ Winkel $< 8\pi$ (Bogenmaß)
$-1600 <$ Winkel < 1600 (Neugrad)
Der Datentyp muß reell mit einfacher Genauigkeit oder reell mit doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktionen geben den trigonometrischen Funktionswert für einen gegebenen Winkel.
SIN: Sinusfunktion
COS: Kosinusfunktion
TAN: Tangensfunktion

(2) Der Datentyp des Funktionswerte entspricht dem Datentyp des gegebenen Winkels.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 ANGLE 0
20 FOR I=0 TO 45 STEP 5
30 PRINT SIN(I)
40 PRINT COS(I)
50 PRINT TAN(I)
60 PRINT
70 NEXT I
```

```
RUN
 0
 1
 0

 8.71557E-02
 .996195
 8.74887E-02

 .173648
 .984808
 .176327

 .258819
 .965926
 .267949
```

5-10-1-3. ASN/ACS/ATN-Funktionen

ASN (Argument)

Numerischer Ausdruck

ACS (Argument)

Numerischer Ausdruck

ATN (Argument)

Numerischer Ausdruck

Funktion

Inverse trigonometrische Funktionen, die einen Winkel angeben, der einem gegebenen Argument entspricht.

Parameter

(1) Argument
Ein numerischer Ausdruck. $-1 \leqq$ Argument $\leqq 1$ (ASN/ACS).
Der Datentyp muß reell mit einfacher Genauigkeit oder reell mit doppelter Genauigkeit sein.

Erklärung

(1) Diese inversen trigonometrischen Funktionen geben einen Winkel, der einem gegebenen Argument entspricht.

ASN: Inverse Sinusfunktion.
ACS: Inverse Kosinusfunktion.
ATN: Inverse Tangensfunktion.

(2) Die Bereiche der Funktionswerte sind wie folgt:

$$-90^\circ \leqq \text{ASN}(X) \leqq 90^\circ,\ 0^\circ \leqq \text{ACS}(X) \leqq 180^\circ,\ -90^\circ \leqq \text{ATN}(X) \leqq 90^\circ$$

(3) Der Datentyp der Funktionswerte ist der gleiche wie der Datentyp der entsprechenden Argumente.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 ANGLE 0
20 FOR I=0 TO 1 STEP .2
30 PRINT ASN(I)
40 PRINT ACS(I)
50 PRINT ATN(I)
60 PRINT
70 NEXT I
```

```
RUN
 0
 90
 0

 11.537
 78.463
 11.3099
 23.5782
 66.4218
 21.8014
 36.8699
 53.1301
```

5-10-2. Logarithmische Funktionen und Exponentialfunktionen

5-10-2-1. LOG/LGT-Funktionen

LOG (Argument)
Numerischer Ausdruck

LGT (Argument)
Numerischer Ausdruck

Funktion

Angabe der Werte logarithmischer Funktionen.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp muß reell mit einfacher Genauigkeit oder reell mit doppelter Genauigkeit sein.

- LOG 0 < **Argumente**
- LGT 0 < **Argumente**

Erklärung

(1) Diese Funktionen geben die Werte logarithmischer Funktionen.

- LOG Natürliche Logarithmusfunktion (Basis e) $\log_e x$、$\ln x$
- LGT Normale Logarithmusfunktion (Basis 10) $\log_{10} x$、$\log x$

(2) Die Datentypen der Funktionswerte sind die gleichen wie die Datentypen der entsprechenden Argumente.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=0 TO 10 STEP 2
20 PRINT I,LOG(EXP(I)),LGT(10^I)
30 NEXT
```

```
RUN
 0   0   0
 2   2   4.60517
 4   4   9.21034
 6   6   13.8155
 8   8   18.4207
 10   10   23.0259
```

5-10-2-2. EXP-Funktion

EXP (Argument)

Numerischer Ausdruck

Funktion

Angabe des Wertes der Exponentialfunktion.

Parameter

(1) Argument
Ein numerischer Ausdruck, dessen Wert kleiner als 230 ist. Der Datentyp muß reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt den Wert der Exponentialfunktion für ein gegebenes Argument.

EXP(x) ..... e^x

(2) Der Datentyp des Funktionswertes ist der gleiche wie der Datentyp des entsprechenden Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=1 TO 5
20 PRINT I,EXP(I)
30 NEXT
```

```
RUN
 1   2.71828
 2   7.38906
 3   20.0855
 4   54.5981
 5   148.413
```

5-10-3. Sonstige arithmetische Funktionen

5-10-3-1. SQR-Funktion

SQR (Argument)

Numerischer Ausdruck

Funktion

Angabe der Quadratwurzel des Arguments

Parameter

(1) Argument: $0 <$ Argument
Ein numerischer Ausdruck, dessen Wert größer als Null ist. Der Datentyp muß reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt die Quadratwurzel des gegebenen Arguments. Der Datentyp der Quadratwurzel ist der gleiche wie der des Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=2 TO 9
20 DBL#=I
30 PRINT I;SQR(DBL#)
40 NEXT
```

```
RUN
 2   1.414213562373095
 3   1.732050807568877
 4   2
 5   2.23606797749979
 6   2.449489742783178
 7   2.645751311064591
 8   2.82842712474619
 9   3
```

5-10-3-2 ABS-Funktion

ABS (Argument)

Numerischer Ausdruck

Funktion

Angabe des absoluten Wertes des Arguments.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt den absoluten Wert eines gegebenen Arguments. Der Datentyp des absoluten Wertes ist der gleiche wie der des Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=-1 TO 1 STEP .4
20 PRINT I,ABS(I)
30 NEXT
```

```
RUN
-1              1
-0.6            0.6
-0.2            0.2
 0.2            0.2
 0.6            0.6
 1              1
```

SGN (Argument)

Numerischer Ausdruck

Funktion

Angabe eines Wertes von −1, 0 oder 1 entsprechend dem Vorzeichen des Arguments.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt einen festen Wert entsprechend dem Vorzeichen des Arguments: −1 für negativ, 0 für 0 und 1 für positiv. Der Datentyp des Wertes ist reell mit einfacher Genauigkeit.

Argument (X)	Vorzeichen (X)
X < 0	− 1
X = 0	0
X > 0	1

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=-2 TO 2
20 PRINT I,
30 ON SGN(I)+1 GOTO 50,60
40 PRINT "(-)":GOTO 70
50 PRINT:GOTO 70
60 PRINT "(+)"
70 NEXT
```

```
RUN
-2              (-)
-1              (-)
 0
 1              (+)
 2              (+)
```

5-10-3-4. INT-Funktion

INT (Argument)

Numerischer Ausdruck

Funktion

Angabe der größten ganzen Zahl, die kleiner als das gegebene Argument oder gleich dem gegebenen Argument ist.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt die größte ganze Zahl, die kleiner als das gegebene Argument oder gleich dem gegebenen Argument ist. Der Datentyp des ganzzahligen Ergebnisses ist der gleiche wie der des Arguments.

Argument	Funktionswert
Ganzzahliger Typ	Reeller Typ einfacher Genauigkeit
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit

Beispiel

```
10 FOR I=-2 TO 2 STEP 0.25
20 PRINT I,INT(I)
30 NEXT I
40 END
```

```
RUN
-2              -2
-1.75           -2
-1.5            -2
-1.25           -2
-1              -1
-0.75           -1
-0.5            -1
-0.25           -1
 0               0
 0.25            0
 0.5             0
 0.75            0
 1               1
 1.25            1
 1.5             1
 1.75            1
 2               2
```

5-10-3-5. FIX-Funktion

FIX (Argument)

Numerischer Ausdruck

Funktion

Angabe des Wertes des ganzzahligen Teils des Arguments

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt den Wert des ganzzahligen Teils des Argumentes an, d.h. sie macht das Argument ganzzahlig. Der Datentyp des Wertes ist der gleiche wie der des Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=-2 TO 2 STEP 0.25
20 PRINT I,FIX(I)
30 NEXT I
40 END
```

```
RUN
-2            -2
-1.75         -1
-1.5          -1
-1.25         -1
-1            -1
-0.75          0
-0.5           0
-0.25          0
 0             0
 0.25          0
 0.5           0
 0.75          0
 1             1
 1.25          1
 1.5           1
 1.75          1
 2             2
```

FRAC (Argument)

Numerischer Ausdruck

Funktion

Angabe des Wertes des Bruchteils des Arguments. Das Vorzeichen des Wertes ist das gleiche wie das des Arguments.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

Erklärung

(1) Diese Funktion gibt den Wert des Bruchteils des Arguments, und der Datentyp des Wertes ist der gleiche wie der des Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=-2 TO 2 STEP 0.25
20 PRINT I,FRAC(I)
30 NEXT I
40 END
```

```
RUN
-2              0
-1.75          -0.75
-1.5           -0.5
-1.25          -0.25
-1              0
-0.75          -0.75
-0.5           -0.5
-0.25          -0.25
 0              0
 0.25           0.25
 0.5            0.5
 0.75           0.75
 1              0
 1.25           0.25
 1.5            0.5
 1.75           0.75
 2              0
```

ROUND (Argument, Stellenposition)

Numerischer Ausdruck — Numerischer Ausdruck

Funktion

Angabe des durch Rundung der bezeichneten Stelle erhaltenen Wertes.

Parameter

(1) Argument
Ein numerischer Ausdruck. Der Datentyp kann reell mit einfacher oder doppelter Genauigkeit sein.

(2) Stellenposition
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$100 >$ Stellenposition > -100

Erklärung

(1) Diese Funktion gibt den durch Rundung der bezeichneten Stelle des Arguments erhaltenen Wert.

(2) Die Ziffer an der nächsthöheren Stelle wird gerundet.

(3) Der Datentyp des Wertes ist der gleiche wie der des Arguments.

Argument	Funktionswert
Reeller Typ einfacher Genauigkeit	Reeller Typ einfacher Genauigkeit
Reeller Typ doppelter Genauigkeit	Reeller Typ doppelter Genauigkeit

Beispiel

```
10 FOR I=-2 TO 2 STEP 0.3
20 PRINT I,ROUND(I,-1)
30 NEXT I
40 END
```

```
RUN
-2              -2
-1.7            -2
-1.4            -1
-1.1            -1
-0.8            -1
-0.5            -1
-0.2             0
 0.1             0
 0.4             0
 0.7             1
 1               1
 1.3             1
 1.6             2
 1.9             2
```

```
10 A#=123456789
20 FOR I=0 TO 7
30 PRINT I,ROUND(A#,I)
40 NEXT
```

```
RUN
 0              123456790
 1              123456800
 2              123457000
 3              123460000
 4              123500000
 5              123000000
 6              120000000
 7              100000000
```

RND (Argument)

Numerischer Ausdruck

Funktion

Angabe einer Zufallszahl zwischen 0 und 1.

Parameter

(1) Argument
Ein numerischer Ausdruck.

Erklärung

(1) Diese Funktion gibt eine Zufallszahl zwischen 0 und 1. Der Datentyp ist reell mit einfacher Genauigkeit ($0 < RND(X) < 1$).

(2) Wenn das Argument positiv ist, so wird eine Zufallszahl aus der Standardfolge gegeben.

(3) Wenn das Argument 0 ist, so wird die gleiche Zufallszahl wie die letzte gegebene Zufallszahl gegeben.

(4) Wenn das Argument negativ ist, so wird eine Zufallszahl aus einer Nichtstandardfolge gegeben.

(5) Bei jedem Programmstart beginnen die Zufallszahlen mit dem gleichen Wert. Zufallszahlen werden deshalb immer in einer bestimmten Reihenfolge erzeugt, außer wenn die RANDOMIZE-Anweisung verwendet wird oder wenn ein negatives Argument für RND gegeben wird.

Beispiel

```
10 A=RND(-1)
20 FOR I=1 TO 10
30 PRINT RND(1)
40 NEXT I
50 END
```

RANDOMIZE

Funktion

Änderung der Zufallszahlenfolge für die RND-Funktion.

5-11. Zeichendatenmanipulation

5-11-1. CHRS-Funktion

CHR$ (Kode)

Numerischer Ausdruck

Funktion

Angabe des durch den gegebenen Kode dargestellten Zeichens.

Parameter

(1) Kode
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Kode < 256

Erklärung

(1) Diese Funktion gibt das durch einen gegebenen Kode dargestellte Zeichen.

Beispiel

```
10 PRINT CHR$(65)
```

```
RUN
A
```

5-11-2. ASC-Funktion

ASC (Zeichenkette)
Zeichenausdruck

Funktion

Angabe des Kodes für das erste Zeichen einer Zeichenkette.

Parameter

(1) Zeichenkette
Ein Zeichenausdruck.

Erklärung

(1) Diese Funktion gibt den Kode, der das erste Zeichen einer Zeichenkette darstellt. Der Wert ist ein reeller Typ einfacher Genauigkeit.

(2) Der Wert 0 wird gegeben, wenn die gegebene Zeichenkette die Lange 0 hat.

Beispiel

```
10 PRINT ASC("AB")
20 END
```

```
RUN
 65
```

5-11-3. STR$-Funktion

STR$ (Argument)

Numerischer Ausdruck

Funktion

Umwandlung des Arguments (einer numerischen Konstanten oder des Wertes eines numerischen Ausdrucks) in eine Zeichenkette von Stellen, Dezimalpunkt usw.

Parameter

(1) Argument
Ein numerischer Ausdruck.

Erklärung

(1) Diese Funktion nimmt einen Wert an, der aus einer Zeichenkette besteht, die durch Umwandlung eines Arguments in Dezimalnotierung erhalten worden ist. Ausdrucken des Wertes der Funktion als Zeichenkette gibt das gleiche Ergebnis wie Ausdrucken des Arguments als numerischer Wert.

Beispiel

```
10 A$="("+STR$(12.5)+")"
20 PRINT A$
30 END
```

```
RUN
( 12.5)
```

11-4. VAL-Funktion

VAL (Zeichenkette)
Zeichenausdruck

Funktion

Annahme des numerischen Wertes eines durch eine aus Ziffern, Dezimalpunkt usw. bestehenden Zeichenkette dargestellten numerischen Ausdrucks.

Parameter

(1) Zeichenkette
Ein Zeichenausdruck

Erklärung

(1) Diese Funktion nimmt den numerischen Wert eines numerischen Ausdrucks an, der durch eine Zeichenkette dargestellt wird, d.h. sie bietet die inverse Funktion für STR$.

(2) Die Darstellung durch eine Zeichenkette muß den gleichen Regeln entsprechen, wie sie für die Darstellung einer numerischen Konstanten gelten.

(3) Wenn in der Darstellung durch die Zeichenkette ein Datentyp bezeichnet ist, so ist der Funktionswert der bezeichnete Datentyp.

(4) Wenn die Darstellung durch eine Zeichenkette nicht den Regeln für die Darstellung einer numerischen Konstanten entspricht, so wird der Wert des Teils der Zeichenkette gegeben, der den Regeln entspricht. Die verbleibenden Zeichen werden nicht beachtet.

(5) Wenn die Zeichenkette die Länge 0 hat oder entsprechend den Regeln für numerische Konstanten kein entsprechender numerischer Wert vorhanden ist, so ist der Wert der Funktion 0 und sie ist vom reellen Typ einfacher Genauigkeit.

Beispiel

```
10 PRINT VAL("123")
20 END
```

```
RUN
 123
```

5-11-5. MIDS-Funktion

MID$ (Zeichenkette, Position [, Anzahl der Zeichen])

Zeichenausdruck Numerischer Ausdruck Numerischer Ausdruck

Funktion

Angabe einer Zeichenkette, bestehend aus der bezeichneten Anzahl von Zeichen von der bezeichneten Position in der gegebenen Zeichenkette.

Parameter

(1) Zeichenkette: Ein Zeichenausdruck

(2) Position
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Position < 256

(3) Anzahl der Zeichen
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Anzahl der Zeichen < 256
Wenn dieser Parameter ausgelassen wird, so werden alle Zeichen von der bezeichneten Position bis zum Ende der Kette eingeschlossen.

Erklärung

(1) Diese Funktion gibt eine Zeichenkette, bestehend aus der bezeichneten Anzahl von Zeichen von der bezeichneten Position in der gegebenen Zeichenkette. Wenn die Anzahl der Zeichen nicht bezeichnet wird, so werden alle Zeichen von der bezeichneten Position an gegeben.

(2) Wenn eine Position uber die Lange der Zeichenkette hinaus bezeichnet wird, so ist der Funktionswert eine Zeichenkette mit der Länge 0.

(3) Wenn die Lange der Zeichenkette von der bezeichneten Position an kürzer als die bezeichnete Anzahl der Zeichen ist, so werden nur die verbleibenden Zeichen eingeschlossen.

Beispiel

```
10 PRINT MID$("123456",2,3)
20 END
```

```
RUN
234
```

5-11-6. LEFTS-Funktion

LEFT$ (Zeichenkette, Anzahl der Zeichen)

Zeichenausdruck — Numerischer Ausdruck

Funktion

Angabe der bezeichneten Anzahl von Zeichen vom linken Ende der Zeichenkette.

Parameter

(1) Zeichenkette: Ein Zeichenausdruck.
(2) Anzahl der Zeichen
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Anzahl der Zeichen < 256

Erklärung

(1) Diese Funktion gibt die bezeichnete Anzahl von Zeichen vom linken Ende der bexeichneten Zeichenkette.

(2) Wenn die Anzahl der Zeichen großer als die Länge der Zeichenkette ist, so wird der Wert der Zeichenkette selbst angenommen.

Beispiel

```
10 PRINT LEFT$("ABCD",3)
20 END
```

```
RUN
ABC
```

5-11-7. RIGHTS-Funktion

RIGHT$ (Zeichenkette, Anzahl der Zeichen)

Zeichenausdruck — Numerischer Ausdruck

Funktion

Angabe der bezeichneten Anzahl von Zeichen vom rechten Ende der Zeichenkette.

Parameter

(1) Zeichenkette: Ein Zeichenausdruck.

(2) Anzahl der Zeichen
Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Anzahl der Zeichen < 256

Erklärung

(1) Diese Funktion gibt die bezeichnete Anzahl von Zeichen vom rechten Ende der bezeichneten Zeichenkette.

(2) Wenn die Anzahl der Zeichen größer als die Länge der Zeichenkette ist, so wird der Wert der Zeichenkette selbst angenommen.

Beispiel

```
10 PRINT RIGHT$("ABCDE",2)
20 END
```

```
RUN
CD
```

5-11-8. LEN-Funktion

LEN (Zeichenkette)
Zeichenausdruck

Funktion

Angabe der Länge der bezeichneten Zeichenkette.

Parameter

(1) Zeichenkette: Ein Zeichenausdruck.

Erklärung

(1) Diese Funktion gibt die Länge einer bezeichneten Zeichenkette an. Das Ergebnis ist vom reellen Typ einfacher Genauigkeit.

Beispiel

```
10 PRINT LEN("ASTERISK")
20 END
```

```
RUN
 8
```

5-12. Funktionsdefinition

5-12-1. DEFFN-Anweisung

DEFFN Funktionsname [(Formalargument / Variablenname [, Formalargument / Variablenname]*)] = Ausdruck

Funktion

Definition einer durch den Benutzer definierten Funktion.

Parameter

(1) Funktionsname

(2) Formalargument — Ein einfacher Variablenname.

(3) Ausdruck — Ein numerischer Ausdruck oder ein Zeichenausdruck.

Erklärung

(1) Diese Anweisung definiert eine durch den Benutzer definierte Funktion.

(2) Dem Funktionsnamen und dem Formalargument können Typbezeichnungssymbole (!, #, $, %) nachgestellt werden, um den Datentyp zu bezeichnen. Wenn der Datentyp nicht bezeichnet ist, so wird der reelle Typ einfacher Genauigkeit angenommen. Wenn der Typ in einer DEF-Anweisung festgelegt ist, so wird der festgelegte Datentyp angewendet.

(3) Das Formalargument wird im Ausdruck auf der rechten Seite verwendet. Es wird separat von externen Variablennamen mit dem gleichen Namen, falls vorhanden, behandelt.

(4) Wenn die Funktion vom Zeichentyp ist, so muß der Ausdruck auf der rechten Seite ein Zeichenausdruck sein, und wenn die Funktion vom numerischen Typ ist, so muß der Ausdruck auf der rechten Seite ein numerischer Ausdruck sein.

(5) Eine durch den Benutzer definierte Funktion kann Bezug auf eine andere durch den Benutzer definierte Funktion nehmen. Beachten Sie die folgenden Punkte:

1) Biz zu fünf Bezugsebenen sind zulässig. (Die Anzahl der Ebenen variiert entsprechend der Komplexität des Ausdrucks.)

2) Eine durch den Benutzer definierte Funktion darf schließlich nicht sich selbst abrufen.

3) Wenn ein Variablenname, ausgenommen ein im Ausdruck verwendetes Formalargument, mit einem auf diese Funktion Bezug nehmenden Argumentnamen übereinstimmt, so wird dieses Argument der Veränderlichen zugeordnet.

5-12-2. FN-Funktionen

FN Funktionsname [(Reelles Argument [, Reelles Argument]*)]
Ausdruck Ausdruck

Funktion

Angabe des Wertes einer im voraus durch die DEFFN-Anweisung definierten Funktion.

Parameter

(1) Funktionsname
(2) Reelles Argument
Ein numerischer Ausdruck oder ein Zeichenausdruck.

Erklärung

(1) Diese Funktion nimmt den Wert einer im voraus durch die DEFFN-Anweisung definierten Funktion an. Der Wert hat den gleichen Datentyp wie die definierte Funktion.
(2) Dem Funktionsnamen und dem Formalargument können Typbezeichnungssymbole (!, #, S.) nachgestellt werden, um den Datentyp zu bezeichene. Wenn der Datentyp nicht bezeichnet ist, so wird der reelle Typ einfacher Genauigkeit angenommen. Wenn der Typ in einer DEF-Anweisung festgelegt ist, so wird der festgelegte Datentyp angewendet.
(3) Die Anzahl der reellen Argumente muß mit der Anzahl der Formalargumente in der Funktionsdefinition übereinstimmen.
(4) Das Formalargument und das entsprechende reelle Argument mussen vom gleichen Datentyp sein. Wenn beide vom numerischen Typ sind, so wird jedoch automatische Umwandlung durchgeführt, um die gleiche Genauigkeit zu erhalten.

Beispiel

```
10 DEFFN A(X,Y)=X*X+Y*Y
20 DEFFN B(X)=X+FN A(X,5)
30 FOR I=0 TO 10 STEP 2
40 PRINT FN B(I)
50 NEXT I
60 END
```

```
RUN
 25
 31
 45
 67
 97
 135
```

5-13. Sonstige Funktionen

5-13-1. FRE-Funktion

FRE (Ausdruck)

Funktion

Angabe der Größe des nicht verwendeten Bereichs für BASIC oder des nicht verwendeten Zeichenbereichs oder des nicht verwendeten CETL-Bereichs.

Parameter

(1) Ausdruck
Ein Zeichenausdruck oder ein numerischer Ausdruck.

Erklärung

(1) Diese Funktion gibt die Größe (in Bytes) des nicht verwendeten Bereichs für BASIC als ihren Wert, wenn der Ausdruck ein numerischer Ausdruck ist, und sie nimmt die Große des nicht verwendeten Zeichenbereichs (in Bytes) als ihren Wert, wenn der Ausdruck ein Zeichenausdruck ist. Der Wert ist vom ganzzahligen Typ.

(2) Wenn der Ausdruck ausgelassen wird, so wird die Größe des nicht verwendeten CETL-Bereichs in Bytes angezeigt.

Beispiel

```
10 CLEAR 1023,&HEFFF
20 PRINT FRE(0),FRE("")
30 A=5
40 PRINT FRE(0),FRE("")
50 A$="0123456"
60 PRINT FRE(0),FRE("")
70 END
```

```
RUN
 18671          1023
 18660          1023
 18655          1015
```

5-13-2. PEEK-Funktion

PEEK (Adresse)

Numerischer Ausdruck

Funktion

Angabe des Wertes des Inhalts einer bezeichneten Speicheradresse.

Parameter

(1) Adresse
Ein als ganze Zahl bewerteter Ausdruck.
$0 \leqslant$ Adresse < 65536

Erklärung

(1) Diese Funktion gibt den Wert des Inhalts der bezeichneten Speicheradresse. Der Funktionswert ist ein reeller Typ einfacher Genauigkeit.

Beispiel

```
10 INPUT "ADDRESS=",AD
20 IF AD<0 OR AD>65536 THEN END
30 PRINT AD,PEEK(AD)
40 GOTO 10
```

```
RUN
ADDRESS=1
 1            16
ADDRESS=2
 2            131
ADDRESS=
```

Siehe in diesem Zusammenhang auch

POKE

5-13-3. POKE-Anweisung

POKE Adresse, Daten

Adresse: Numerischer Ausdruck
Daten: Numerischer Ausdruck

Funktion

Schreiben von Daten in eine bezeichnete Adresse.

Parameter

(1) Adresse
Ein als ganze Zahl bewerteter Ausdruck.
$0 \leqslant$ Adresse < 65536

(2) Daten
Ein als ganze Zahl bewerteter Ausdruck.
$0 \leqslant$ Daten < 256

Erklärung

(1) Durch diese Anweisung werden die bezeichneten ganzzahligen Daten in den bezeichneten Speicher geschrieben.

Beispiel

```
 10 INPUT "ADDRESS=",AD
 20 IF AD<0 OR AD>65536 THEN END
 30 PRINT AD;PEEK(AD);
 40 DT$=INKEY$:IF DT$="" THEN 40
 50 IF DT$=CHR$(13) THEN END
 60 IF DT$=" " THEN AD=AD+1:PRINT:GOTO 30
 70 IF DT$=CHR$(30) THEN AD=AD-1:PRINT:GOTO 30
 80 PRINT DT$:DT=VAL(DT$)
 90 POKE AD,DT:AD=AD+1
100 GOTO 30
```

Siehe in diesem Zusammenhang auch

PEEK-Anweisung

DATE$

TIME$

Funktion

Wiedergabe des Datums bzw. der Zeit.

Erklärung

(1) Der Speicherinhalt wird durch die Stutzbatterie erhalten. Die Funktionen DATES und TIMES verwenden diese Eigenschaft, um die Zeit bzw. das Datum anzugeben.

(2) DATES wird im Format "YY/MM/DD" und TIMES wird im Format "HH:MM:SS" angezeigt.

(3) DATES und TIMES konnen durch Zuweisungsanweisungen eingestellt werden. Fur beide Funktionen kann ein beliebiges Abgrenzungszeichen fur die dritte und die sechste Stelle von links eingegeben werden. Zeichen nach der neunten Stelle werden nicht beachtet.

(4) Die Werte von DATES und TIMES bleiben auch bei Durchführung der RESET-Anweisung erhalten.

Beispiel

```
10 DATES="83 01 26" : TIMES="19 : 45 : 58"
20 PRINT DATES, TIMES
```

6. STATISTISCHE VERARBEITUNG

Durch Verwendung der folgenden Befehle und Funktionen können grundlegende statistische Informationen leicht verarbeitet werden, und lineare Regressionsanalyse kann durchgeführt werden.

6-1. STAT-CLEAR-Befehl

STAT CLEAR

Funktion

Initialisieren der statistischen Verarbeitungsfunktionen. Dieser Befehl muß vor statistischer Verarbeitung durchgeführt werden.

STAT X-Daten [,Y-Daten]

Numerischer Ausdruck Numerischer Ausdruck

Funktion

Eingabe statistischer Daten.

Parameter

(1) X-Daten — Ein numerischer Ausdruck, dessen Wert in einen reellen Typ doppelter Genauigkeit umgewandelt wird.

(2) Y-Daten — Ein numerischer Ausdruck, dessen Wert in einen reellen Typ doppelter Genauigkeit umgewandelt wird. Wenn dieser Parameter ausgelassen wird, so nimmt das System den Wert der vorhergehenden Y-Daten an.

Erklärung

(1) Durch diesen Befehl wird statistische Verarbeitung von X- und Y-Daten durchgeführt.

Beispiel

```
 10 READ N
 20 STAT CLEAR
 30 FOR I=1 TO N
 40 READ X,Y
 50 STAT X,Y
 60 NEXT I
 70 PRINT "mean of x";MEANX
 80 PRINT "mean of y";MEANY
 90 PRINT "y=";LRA;"+";LRB;"*x"
100 END
1000 DATA 10
1010 DATA 3.5, 2.4, 12.3,15.4, 49,18, 25,27
1020 DATA 4.7, 4.9, 14,17, 22.3,28, 49,19
```

```
RUN
mean of x 18.46
mean of y 14.94
y= 9.425881718774064
 + .2987062990913291
 *x
```

6-3. Statistische Funktionen

Funktion

Dies sind Funktionen, die die Ergebnisse statistischer Verarbeitung als ihren Wert annehmen. Der Datentyp ist immer reell mit doppelter Genauigkeit.

Erklärung

$$Y = \underset{LRb}{b}x + \underset{LRA}{a}$$

Funktions-name	Wert	Ausdruck
CNT	Anzahl der Eingabedatenpunkte für statistische Verarbeitung	n
SUMX	Summe der X-Daten	Σx
SUMY	Summe der Y-Daten	Σy
SUMX 2	Summe der Quadrate der X-Daten	Σx^2
SUMY 2	Summe der Quadrate der Y-Daten	Σy^2
SUMXY	Summe der Produkte der X- und Y-Daten	Σxy
MEANX	Mittelwert der X-Daten	$\frac{\Sigma x}{n}$
MEANY	Mittelwert der Y-Daten	$\frac{\Sigma y}{n}$
SDX	Probenstandardabweichung der X-Daten	$\sqrt{\frac{n\Sigma x^2-(\Sigma x)^2}{n(n-1)}}$
SDY	Probenstandardabweichung der Y-Daten	$\sqrt{\frac{n\Sigma y^2-(\Sigma y)^2}{n(n-1)}}$
SDXN	Standardabweichung von der Gesamtheit der X-Daten	$\sqrt{\frac{n\Sigma x^2-(\Sigma x)^2}{n^2}}$
SDYN	Standardabweichung von der Gesamtheit der Y-Daten	$\sqrt{\frac{n\Sigma y^2-(\Sigma y)^2}{n^2}}$
LRA	Koeffizient für lineare Regression	$\frac{\Sigma y - LRB \cdot \Sigma x}{n}$
LRB	Konstantenausdruck für lineare Regression	$\frac{n\Sigma xy-\Sigma x\cdot\Sigma y}{n\Sigma x^2-(\Sigma x)^2}$
SY	gemittelte Abweichung y	$\sqrt{\frac{\Sigma(y_i-y_j)^2}{N-2}}$
SX	-1- x	$\sqrt{cnt}\cdot SDXN$
Sb	Fehler der Geardensteigung	$\frac{S_y}{S_x}$
Sa	Fehler des Startwertes	$S_y\cdot\sqrt{cnt-\frac{(\Sigma x)^2}{\Sigma x^2}}$

7. Grafiken

7-1. INT-Anweisung

INIT (X-Koordinate, Y-Koordinate), X-Zuwachs, Y-Zuwachs

Numerischer Ausdruck — Numerischer Ausdruck — Numerischer Ausdruck — Numerischer Ausdruck

Funktion

Erstellen von Benutzerkoordinaten in Begriffen absoluter Koordinaten.

Parameter

(1) X-Koordinate — Ein als ganze Zahl bewerteter numerischer Ausdruck. Dies ist eine absolute Koordinate.
$0 \leqslant$ X-Koordinate $<$ 160 (in 80-Zeichen-Betriebsart)

(2) Y-Koordinate — Ein als ganze Zahl bewerteter numerischer Ausdruck. Dies ist eine absolute Koordinate.
$0 \leqslant$ Y-Koordinate $<$ 64 (in Bildschirmbetriebsart 0 oder 2)

(3) X-Zuwachs — Ein numerischer Ausdruck. Der X-Zuwachs darf nicht 0 sein.

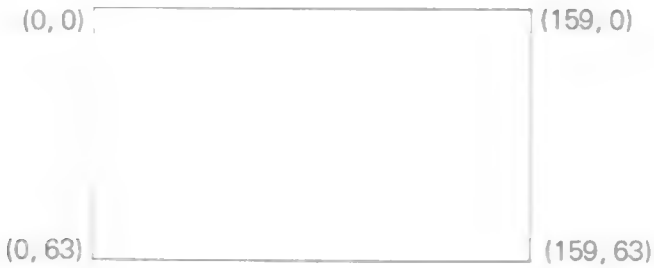
(4) Y-Zuwachs — Ein numerischer Ausdruck. Der Y-Zuwachs darf nicht 0 sein.

Erklärung

Unter Verwendung von durch absolute Koordinaten gegebenen X- und Y-Koordinate als Ursprung erstellt diese Anweisung Koordinaten, in denen X-Zuwachs die Vergrößerung in der X-Richtung und Y-Zuwachs die Vergrößerung in der Y-Richtung darstellt.

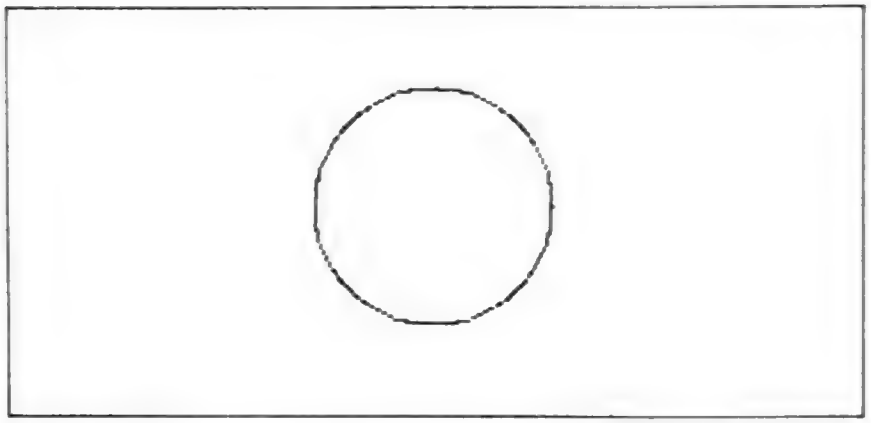
Beispiel

Die Koordinaten sind wie folgt:



Beispiel

```
10 CLS
20 INIT(79,31),30,30:ANGLE 0
30 FOR I=0 TO 360
40 DRAW(COS(I),SIN(I))
50 NEXT I
60 END
```



7-2. DRAW/DRAWC-Anweisungen

{DRAW / DRAWC}	[–]	(X,	Y)	[{:} (X,	Y)]*
		Numerischer Ausdruck	Numerischer Ausdruck	Numerischer Ausdruck	Numerischer Ausdruck

Funktion

Zeichnen von Punkten und Geraden auf der Flüssigkristallanzeige.

Parameter

(1) X: Ein numerischer Ausdruck. Eine Benutzerkoordinate

(2) Y: Ein numerischer Ausdruck. Eine Benutzerkoordinate

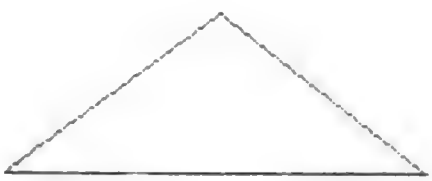
Erklärung

(1) Wenn ein Minuszeichen (–) vor (X, Y) steht, so wird durch diese Anweisung eine Gerade vom vorher angezeigten Punkt zum durch (X, Y) bezeichneten Punkt gezogen. Wenn kein Minuszeichen (–) vor (X, Y) steht, so wird der durch (X, Y) festgelegte Punkt angezeigt.

(2) Die DRAW-Anweisung zeichnet die festgelegten Punkte bzw. Geraden, und die DRAWC-Anweisung löscht sie.

Beispiel

```
 10 INIT(0,0),1,1:CLS
 20 FOR I=1 TO 4
 30 READ X(I),Y(I)
 40 NEXT I
 50 DRAW(X(1),Y(1))
 60 FOR I=2 TO 4
 70 DRAW-(X(I),Y(I))
 80 NEXT I
 90 FOR I=2 TO 4
100 DRAWC-(X(I),Y(I))
110 NEXT I
120 END
```



QUAD / QUADC [(X,Y)] -(X,Y)

Funktion

Zeichnen von Vierecken.

Parameter

(1) X — Ein numerischer Ausdruck. Eine durch den Benutzer definierte Koordinate.

(2) Y — Ein numerischer Ausdruck. Eine durch den Benutzer definierte Koordinate.

Erklärung

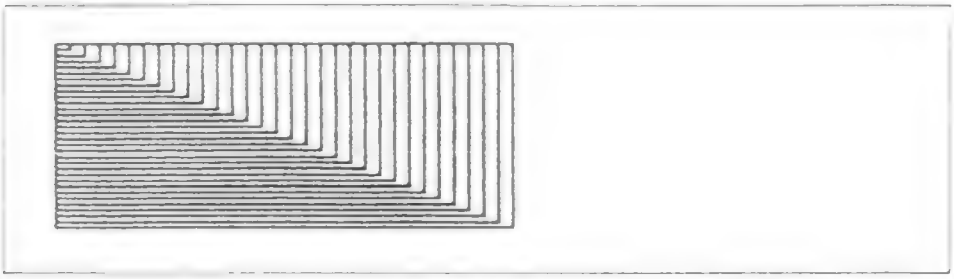
(1) Durch diese Anweisung werden Rechtecke mit zwei festgelegten Punkten als ihren Eckpunkten gezeichnet.

(2) Wenn ein Punkt ausgelassen wird, so wird an seiner Stelle der zuletzt gezeichnete Punkt verwendet.

(3) Mit der QUADC-Anweisung wird das spezifizierte Rechteck gelöscht; die Spezifikation erfolgt in der gleichen Weise wie bei der QUAD-Anweisung.

Beispiel

```
10 INIT(0,0),5,2:CLS
20 FOR I=0 TO 31
30 QUAD(0,0)-(I,I)
40 NEXT
```



POINT (X , Y)

Numerischer Ausdruck — Numerischer Ausdruck

Funktion

Diese Funktion wird verwendet, um zu prüfen, ob ein gegebener Punkt ein- oder ausgeschaltet ist.

Parameter

(1) X Ein numerischer Ausdruck. Durch den Benutzer definierte Koordinaten.
(2) Y Ein numerischer Ausdruck. Durch den Benutzer definierte Koordinaten.

Erklärung

(1) Diese Funktion nimmt den Wert –1 für den festgelegten Koordinatenpunkt an, wenn der Punkt eingeschaltet ist, und sie nimmt den Wert 0 an, wenn der Punkt ausgeschaltet ist. Das Ergebnis ist vom reellen Typ einfacher Genauigkeit.

Beispiel

```
 10 CLS:DEFSNG A-Z:CLEAR
 20 DIM A(15,7)
 30 PRINT "FP"
 40 FOR Y=0 TO 7
 50 FOR X=0 TO 15
 60 A(X,Y)=POINT(X,Y)
 70 NEXT X,Y
 80 CLS
 90 FOR Y=0 TO 7
100 PRINT
110 FOR X=0 TO 15
120 IF A(X,Y) THEN A$="■" ELSE A$=" "
130 PRINT A$;
140 NEXT X,Y
```

8. DRUCKERSTEUERUNG

8-1. LLIST-Anweisung

LLIST [Startzeilennummer] [{-, ,} [Endzeilennummer]]

Zeilennummer Zeilennummer

Funktion

Drucken des Inhalts eines Programms auf einem Drucker.

Parameter

(1) Startzeilennummer Die Nummer der ersten zu druckenden Zeile.
* Wenn dieser Parameter ausgelassen wird, so wird der Anfang des Programas angenommen.

(2) Endzeilennummer Die Nummer der letzten zu druckenden Zeile.
* Wenn diese Nummer ausgelassen wird, so wird das Ende des Programms angenommen.

Startzeilennummer und Endzeilennummer sind Zeilennummen im Bereich von 1 bis 64999 .

Erklärung

(1) Durch diese Anweisung werden die bezeichneten Zeilennummern auf dem Drucker ausgedruckt.

(2) Der Unterschied zuf LIST-Anweisung liegt darin, daß durch diese Anweisung der Inhalt eines Programms nicht auf der Flüssigkristallanzeige angezeigt wird. Beziehen Sie sich für weitere Informationen auf den Abschnitt unter "LIST-Anweisung".

Beispiel

LLIST 10, 100

In Zusammenhang stehender Punkt

LIST

8-2. LPRINT-Anweisung

LPRINT [Ausgabeelement] [{;/,} [Ausgabeelement]]*

Ausgabeelement { TAB (numerischer Ausdruck) / Numerischer Ausdruck / Zeichenausdruck }

Funktion

Ausgabe von Zeichen zum Drucker.

Parameter

(1) Ausgabeelement — Eine Ausgabesteuerfunktion oder ein numerischer Ausdruck oder ein Zeichenausdruck.

Erklärung

(1) Ausgabe eines Ausgabeelements zum Drucker. Wenn das Ausgabeelement eine Ausgabesteuerfunktion ist, so wird die durch die Funktion bezeichnete Operation ausgeführt. Wenn das Ausgabeelement ein numerischer Ausdruck oder ein Zeichenausdruck ist, so wird der Wert des Ausdrucks zum Drucker ausgegeben.

(2) Die Werte für numerische Ausdrücke werden in Dezimalnotierung im gleichen Format wie das der PRINT-Anweisung gedruckt (siehe Abschnitt " PRINT-Anweisung").

(3) Die Werte von Zeichenausdrücken werden direkt zum Drucker ausgegeben.

(4) Wenn Ausgabeelemente durch Kommas (,) getrennt sind, so wird eine Zonentabulierung vor Ausgabe des auf das Komma folgenden Ausgabeelements durchgeführt. Ein Zonentabulator ist alle 10 Zeichen gesetzt, mit Zählung von 0 bis 255 Zeichen. Durch Zonentabulierung werden Leerstellen von der gegenwärtigen Position bis zur nächsten Zonentabulatorposition ausgegeben. Ein auf ein Komma (,) folgendes Ausgabeelement wird also vom Anfang einer Zonentabulierung an gedruckt.

```
10 LPRINT
20 FOR I=1 TO 20:LPRINT "*";I,:NEXT I
30 LPRINT
40 END
```

```
* 1          * 2          * 3          * 4          * 5          * 6          * 7          * 8
* 9          * 10         * 11         * 12         * 13         * 14         * 15         * 16
* 17         * 18         * 19         * 20
```

(5) Wenn Ausgabeelemente durch Semikolon (;) getrennt sind, so wird das auf das Semikolon folgende Ausgabeelement unmittelbar nach dem Ausgabeelement vor dem Semikolon gedruckt.

```
10 LPRINT
20 FOR I=1 TO 50
30 LPRINT "(";I;")";
40 NEXT I
50 LPRINT
60 END
```

```
( 1 )( 2 )( 3 )( 4 )( 5 )( 6 )( 7 )( 8 )( 9 )( 10 )( 11 )( 12 )( 13 )( 14 )( 15
)( 16 )( 17 )( 18 )( 19 )( 20 )( 21 )( 22 )( 23 )( 24 )( 25 )( 26 )( 27 )( 28 )(
29 )( 30 )( 31 )( 32 )( 33 )( 34 )( 35 )( 36 )( 37 )( 38 )( 39 )( 40 )( 41 )( 4
2 )( 43 )( 44 )( 45 )( 46 )( 47 )( 48 )( 49 )( 50 )
```

(6) Ausgabeelemente können durch Leerstellen getrennt sein. Es ist sogar möglich, ein Abgrenzungszeichen auszulassen, solange die einzelnen Ausgabeelemente unzweideutig unterschieden werden konnen. In beiden Fällen wird die gleiche Operation wie bei Verwendung eines Semikolons (;) durchgeführt.

(7) Wenn die Anweisung mit einem Semikolon (;) endet, so bleibt die Position unmittelbar nach dem letzten gedruckten Zeichen als die nächste Druckposition festgelegt.

(8) Wenn die Anweisung mit einem Komma (,) endet, so wird nach dem Drucken des letzten Ausgabeelements eine Zonentabulierung durchgeführt.

(9) Wenn das Ende der Anweisung weder ein Semikolon (;) noch ein Komma (,) ist, so wird nach dem Drucken des letzten Ausgabeelements ein Zeilenvorschub durchgeführt. In diesem Fall wird die gegenwärtige Druckposition auf Null eingestellt. Die Druckposition wird von 0 bis 255 gezählt und beginnt dann wieder bei 0. Zonentabulierung und TAB-Funktion werden entsprechend der Druckpositionszählung durchgefuhrt. Wenn die Zahlung auf 0 zurückgestellt wird, so wird CR-LF (interne Kodes 0D, 0A) ausgegeben.

(10) Der Drucker beginnt zu drucken, nachdem ein Zeilenvorschub durchgeführt worden ist. Nachdem der Drucker am rechten Rand des Formulars druckt, wird automatisch ein Zeilenvorschub durchgeführt.

TAB (Tabulatorspezifikation)

numerischer Ausdruck

Funktion

Drucken von Leerstellen bis zur bezeichneten Druckposition.

Parameter

(1) Tabulatorspezifikation: Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Tabulatorspezifikation < 256

Erklärung

(1) Diese Funktion wird in der LPRINT-Anweisung verwendet, um Leerstellen auf dem Drucker zu drucken, bis die Druckposition mit der Tabulatorspezifikation übereinstimmt.

(2) Wenn die Tabulatorspezifikation kleiner als die gegenwärtige Druckposition ist, so wird ein Zeilenvorschub durchgeführt, um die Druckposition auf 0 zurückzustellen, und dann werden Leerstellen bis zur bezeichneten Druckposition gedruckt.

(3) Wenn die Druckposition 256 erreicht, so wird sie für weitere Zählung auf 0 zurückgestellt.

Beispiel

```
10 ' TAB(*)
20 LPRINT
30 DATA 20,50,40,20,50,40
40 FOR I=1 TO 6:READ TBCOUNT:LPRINT TAB(TBCOUNT); "*";TBCOUNT;:NEXT I
50 LPRINT
60 END
```

```
* 20                    * 50
                * 40
* 20                    * 50
                * 40
```

LPRINT USING Formatspezifikation ; Ausgabeelement [{;} Ausgabeelement]

[{;}]

Funktion

Drucken von Ausgabeelementen auf dem Drucker in einem bezeichneten Format.

Parameter

(1) Formatspezifikation — Eine aus mindestens einem Zeichen bestehende Zeichenkette.

(2) Ausgabeelement — Ein numerischer Ausdruck oder ein Zeichenausdruck.

Erklärung

(1) Durch diese Anweisung werden Ausgabeelemente auf dem Drucker im bezeichneten Format gedruckt. Beziehen Sie sich für eine eingehende Beschreibung der Formatspezifikationen auf den Abschnitt unter "PRINT-USING-Anweisung".

Siehe in diesem Zusammenhang auch

PRINT USING

Beispiel

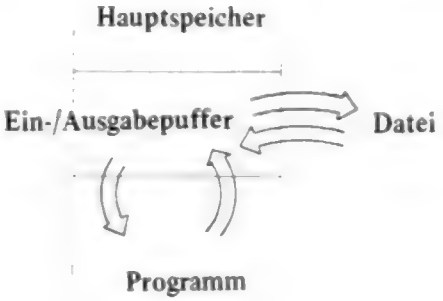
Beziehen Sie sich auf den Abschnitt "PRINT-USING-Anweisung".

9. DATEIVERARBEITUNG

9-1. Sequentielle Verarbeitung und Direktverarbeitung

Die Verarbeitung von Datenpunkten in der Reihenfolge ihres Auftretens wird sequentielle Verarbeitung genannt, während andererseits die Verarbeitung von Datenpunkten in einer willkurlichen Reihenfolge Direktverarbeitung genannt wird. Auf die gleiche Weise konnen Dateien in sequentielle und Direktzugriffsdateien klassifiziert werden. Sequentielle Dateien werden normalerweise durch sequentielle Verarbeitung verarbeitet. Ein Datenpunkt in einer Datei wird als Datensatz bezeichnet. Datenpunkte (Datensätze) in einer sequentiellen Datei können von variabler Länge sein, und sie werden durch einen CRLF-Kode (0D, 0A) getrennt.
Direktzugriffsdateien werden normalerweise durch Direktverarbeitung verarbeitet. Datenzätze in einer Direktzugriffsdateien sind von fester Länge (256 Bytes), und Zugriff erfolgt durch ihre Identifizierungsnummern (Datensatznummern). Direktzugriffsdateien werden nur für FDDs (Floppy Disk Laufwerke) unterstützt.

Grundlagen der Dateiverarbeitung



Datenaustausch zu oder von einer Datei erfolgt über einen Ein-/Ausgabepuffer genannten zeitweiligen Speicherbereich im Speicher. Durch die MOUNT-Anweisung können bis zu 15 Ein-/Ausgabepuffer im Hauptspeicher reserviert werden. Die Anzahl der Ein-/Ausgabepuffer stellt die Anzahl der Dateien dar, die gleichzeitig verarbeitet werden können. Die Ein-/Ausgabepuffer #1 bis #15 werden bestimmten Dateien durch die OPEN-Anweisung zugeordnet. Zusätzlich zu diesen Ein-/Ausgabepuffern ist ein besonderer Ein-/Ausgabepuffer #0 vorhanden, der durch das C_{85}-BASIC-Auswerteprogramm verwendet wird.

Führen Sie zur Verwendung einer Datei das nachfolgend beschriebene Verfahren durch.

(1) Eröffnen Sie die Datei durch die OPEN-Anweisung, und legen Sie entweder sequentielle Verarbeitung oder Direktverarbeitung fest. Wenn sequentielle Verarbeitung bezeichnet wird, so legen Sie fest, ob die Datei eine Eingabedatei oder eine Ausgabedatei ist. Zu diesem Zeitpunkt wird die Datei einem Ein-/Ausgabepuffer zugeordnet. Während der Verarbeitung der Datei wird die Ein-/Ausgabepuffernummer als die Dateinummer verwendet, und sie identifiziert die Datei eindeutig, bis die Datei durch eine CLOSE-Anweisung geschlossen wird.

(2) Verarbeiten Sie die Datei durch Dateiverarbeitungsanweisungen.

(3) Erklären Sie das Ende der Verarbeitung der Datei durch die CLOSE-Anweisung.

9-1-1. MOUNT-Anweisung

MOUNT Pufferanzahl

numerischer Ausdruck

Funktion

Festlegung der Anzahl der im Speicher zu reservierenden Dateipuffer.

Parameter

(1) Pufferanzahl — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$0 \leqslant$ Pufferanzahl < 16

Erklärung

(1) Diese Anweisung erstellt die bezeichnete Anzahl von Dateipuffern.

(2) Für jede Datei ist ein Dateipuffer erforderlich. Eine Datei wird einem Dateipuffer durch die OPEN-Anweisung zugeordnet, und sie wird nur verwendet, wenn sie offen ist. Es ist deshalb moglich, einen Dateipuffer für mehr als eine Datei zu verwenden, wenn diese nicht gleichzeitig verwendet werden (offen sind). Aus diesem Grund ist es ausreichend, die Anzahl von Dateien als die Anzahl der zu reservierenden Puffer festzulegen, die maximal gleichzeitig verwendet werden.

(3) Diese Anweisung kann nicht in einem Programm durchgeführt werden.

(4) Nach Durchführung dieser Anweisung sind alle Variablen gelöscht.

(5) Beim Versand ab Werk bzw. nach Durchführung eines RESET-Befehls ist die Anzahl der Puffer auf 0 eingestellt. Führen Sie in diesem Fall vor Dateiverarbeitung einen MOUNT-Befehl in direkter Betriebsart durch. Die vorherige Anzahl der Stromversorgung erhalten.

(6) Wenn dieser Befehl ausgeführt wird, so werden alle offenen Dateien geschlossen.

(7) Jeder durch die MOUNT-Anweisung eingestellte Puffer erfordert 316 Bytes des Speichers. Dieser Speicherbereich wird nicht im BASIC-Bereich, sondern im CETL-Bereich reserviert.

OPEN Dateideskriptor [FOR {INPUT / OUTPUT}] AS [#] Dateinummer

Funktion

Erklärung des Anfangs der Dateiverarbeitung.

Parameter

(1) Dateideskriptor — Ein Zeichenausdruck.

(2) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Diese Anweisung ermöglicht die Verwendung einer durch den Dateideskriptor bezeichneten Datei.

(2) Sie ordnet der Datei weiterhin einen durch die Dateinummer bezeichneten Ein-/Ausgabepuffer zu. In der nachfolgenden Verarbeitung wird die Dateinummer für Bezug auf die Datei verwendet.

(3) Wenn FOR INPUT festgelegt wird, so ist die Datei eine Eingabedatei für sequentielle Verarbeitung. Fur FDDs verursacht Bezeichnung einer nicht existierenden Datei einen Fehler.

(4) Wenn FOR OUTPUT festgelegt wird, so ist die Datei eine Ausgabedatei für sequentielle Verarbeitung. Fur FDDs wird eine neue Datei erzeugt. Wenn eine Datei mit dem gleichen Namen schon existiert, so wird sie überschrieben. Wenn die nachfolgend geschriebene Datei kleiner als die ursprüngliche Datei ist, so wird EOF-Anzeige nicht aktualisiert. Führen Sie deshalb zuerst KILL durch, um die vorhandene Datei zu loschen, wenn ein ganzlich verschiedener Inhalt in die Datei geschrieben werden soll.

(5) Wenn weder FOR INPUT noch FOR OUTPUT festgelegt wird:

1) FDD (0:)

Frei wählbare Verarbeitung ist festgelegt. Wenn eine Datei schon vorhanden ist, so wird diese Datei festgelegt. Wenn keine Datei vorhanden ist, so wird für das FDD eine neue Datei erzeugt.

2) Datenübertragungsleitungen (COM 0:)

Es wird eine Ein-/Ausgabedatei für sequentielle Verarbeitung festgelegt.

(6) Wenn versucht wird, eine schon offene Datei zu öffnen, so wird ein Fehler verursacht.

```
10 OPEN "SINTB" FOR OUTPUT AS #1
20 FOR I=0 TO 360 STEP 10
30 PRINT #1,I;",";SIN(I)
40 NEXT I
50 CLOSE #1
60 END
```

9-1-3. CLOSE-Anweisung

```
CLOSE [ [ # ] Dateinummer ] [ , [ # ] Dateinummer ] *
      numerischer Ausdruck   numerischer Ausdruck
```

Funktion

Schließen einer Datei und Erklaren des Endes der Verwendung des Ein-/Ausgabepuffers.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Durch diese Anweisung werden die durch die Dateinummern bezeichneten Dateien geschlossen.

(2) Wenn keine Dateinummer bezeichnet wird, so werden alle gegenwärtig offenen Dateien geschlossen.

(3) Schließen von Dateien, die nicht geoffnet worden sind, verursacht keinen Fehler.

9-2. Sequentielle Verarbeitung

Sequentielle Verarbeitung handhabt sequentielle Dateien durch sequentielle Verarbeitung der Dateidaten vom Anfang an. Die OPEN-Anweisung bezeichnet den Anfang einer sequentiellen Datei und legt weiterhin fest, ob es sich um eine Eingabedatei oder eine Ausgabedatei handelt.

Eine sequentielle Datei besteht aus Datensatzen mit variabler Länge, die durch interne Kodes 0D, 0A getrennt sind. Am Ende einer Datei wird ein SUB-Zeichen (interner Kode 1A) angehängt (für FDDs).

Sequentielle Dateikonfiguration

Sequentielle Dateien sind entweder Eingabedateien für Dateneingabe oder Ausgabedateien für Datenausgabe.

Die zulässigen Ein-/Ausgabespezifikationen variieren entsprechend dem Gerätetyp.

GERÄTNAME	OFFEN FÜR EINGABE	OFFEN FÜR AUSGABE	OFFEN
Kassettentonbandgerät (CAS0 :)	○	○	×
Floppy Disk Laufwerk (0:)	○	○	○
Datenubertragungsleitung (COM 0 :)	○	○	×

× Nicht erlaubt
○: Erlaubt

Eingabeanweisungen können für Eingabedateien verwendet werden und Ausgabeanweisungen für Ausgabedateien. Eine Funktion namens EOF kann verwendet werden, um das Ende einer Eingabedatei zu entdecken (für Floppy Disk Laufwerk und Kassettentonbandgeräte).

ANWEISUNG/FUNKTION	OFFEN FÜR EINGABE	OFFEN FÜR AUSGABE
PRINT #	×	○
PRINT # USING	×	○
INPUT #	○	×
EOF	○	△

△: Nicht zutreffend
x: Nicht erlaubt
o: Erlaubt

Nachfolgend ist ein typisches Programm gezeigt. Durch dieses Programm wird der Inhalt einer bezeichneten Datei zur RS232C ausgegeben.

MOUNT 2 ↵ (Diesen Befehl in Direktbetriebsart ausführen)

```
10 OPEN "0:DATA" FOR INPUT AS #1
20 OPEN "COM0:" FOR OUTPUT AS #2
30 IF EOF(1) THEN CLOSE:END
40 INPUT #1,A$:PRINT A$;
50 PRINT #2,A$
60 GOTO 30
```

9-2-1. PRINT#-Anweisung

PRINT # Dateinummer [, Ausgabeelement[;/, [Ausgabeelement]]*]

Dateinummer: numerischer Ausdruck

★ Ausgabeelement: { TAB( ) / Zeichenausdruck / numerischer Ausdruck }

Funktion

Ausgabe von Daten zu einer sequentiellen Datei.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Durch diese Anweisung werden die Ausgabeelemente sequentiell zu der durch die Dateinummer bezeichneten sequentiellen Datei ausgegeben.

(2) Der Inhalt der Ausgabe ist der gleiche wie bei Ausgabe zu einem Drucker bei Verwendung der LPRINT-Anweisung. Beziehen Sie sich auf den Abschnitt unter "LPRINT#".

(3) Wenn das Ende der Anweisung weder ein Semikolon (;) noch ein Komma (,) ist, so wird CRLF (interne Kode 0D, 0A) am Ende der Datenausgabe ausgegeben.

(4) Diese Anweisung ist nur für Ausgabeanweisungen für sequentielle Verarbeitung gültig.

Beispiel

```
10 ' SAMPLE OF PRINT #n
20 INPUT "FILE NAME";F$
30 OPEN F$ FOR OUTPUT AS #1
40 FOR I=0 TO 10
50 PRINT #1,I
60 PRINT #1,SQR(I)
70 NEXT I
80 CLOSE #1
90 END
```

9-2-2. PRINT#USING-Anweisung

Funktion

Ausgabe von Ausgabeelementen im bezeichneten Format zur sequentiellen Datei.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck. $1 \leqslant$ Dateinummer < 16

(2) Formatspezifikation: Ein Zeichenausdruck.

(3) Ausgabeelement — Ein numerischer Ausdruck oder Zeichenausdruck.

Erklärung

(1) Durch diese Anweisung werden die Ausgabeelemente entsprechend der Formatspezifikation zu der durch die Dateinummer bezeichneten Datei ausgegeben.

(2) Beziehen Sie sich für die Bedeutung der Formate auf den Abschnitt "PRINT-USING-Anweisung".

(3) Diese Anweisung ist nur für Ausgabedateien für sequentielle Verarbeitung gültig.

Siehe in diesem Zusammenhang auch

PRINT USING

9-2-3. INPUT#-Anweisung

INPUT # Dateinummer, Variablenname [, Variablenname] *

numerischer Ausdruck

Funktion

Lesen von Daten aus einer sequentiellen Datei.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

(2) Variablenname

Erklärung

(1) Durch diese Anweisung werden Daten von der durch die Dateinummer bezeichneten Datei gelesen.

(2) Die gelesenen Daten müssen in dem gleichen Format erscheinen wie die durch die INPUT-Anweisung eingegebenen Daten. Daten können also durch Kommas (,), oder CRLF-Zeichen (interne Kodes 0D, 0A) getrennt werden. Die internen Kodes 00 bis 1F können nicht eingegeben werden.

(3) Diese Anweisung ist nur für Eingabedateien mit sequentieller Verarbeitung gültig.

Beispiel

```
10 OPEN "SINTB" FOR INPUT AS #1
20 IF EOF(1) THEN CLOSE:END
30   INPUT #1,A,B
40   PRINT I,J
50 GOTO 20
60 END
```

9-2-4. EOF-Funktion

EOF **(Dateinummer)**

numerischer Ausdruck

Funktion

Anzeige ob das Ende einer Datei gefunden worden ist.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Diese Funktion zeigt das Ende einer Datei an, die gelesen wird. Normalerweise nimmt diese Funktion den Wert 0 an. Nach dem Lesen des letzten Datensatzes nimmt sie den Wert −1 an. Der Datentyp ist von einfacher Genauigkeit.

(2) Wenn bei sequentieller Dateiausgabe eine Datei mit dem gleichen Namen schon vorhanden ist und eine kleinere Datei als die vorhergehende zu schreiben ist, so kann die EOF-Anzeige nicht aktualisiert werden.

Beispiel

```
 10 'EOF SAMPLE
 20 '---DATA OUT---
 30 OPEN "SAMPLE" FOR OUTPUT AS #1
 40 FOR I=0 TO 360 STEP 30
 50 PRINT #1,I;",";SIN(I)
 60 NEXT
 70 CLOSE #1
 80 '--READ START--
 90 OPEN "SAMPLE" FOR INPUT AS #1
100 PRINT "--- READ START ---"
110 IF EOF(1) THEN 150
120  INPUT #1,A,S
130  PRINT A,S,"(EOFF=";EOF(1);")"
140 GOTO 110
150 PRINT "--- READ END ---"
160 PRINT
170 CLOSE #1
180 END
```

```
RUN
--- READ START ---
 0            0
(EOFF= 0 )
 30           0.5
(EOFF= 0 )
 60           .866025
(EOFF= 0 )
 90           1
(EOFF= 0 )
 120          .866025
(EOFF= 0 )
 150          0.5
(EOFF= 0 )
 180          0
(EOFF= 0 )
 210         -0.5
(EOFF= 0 )
 240         -.866025
(EOFF= 0 )
 270         -1
(EOFF= 0 )
 300         -.866025
(EOFF= 0 )
 330         -0.5
(EOFF= 0 )
 360          0
(EOFF=-1 )
--- READ END ---
```

9-3. Direktverarbeitung

Direktverarbeitung erfolgt für Direktzugriffsdateien. Durch Datensatznummern bezeichnete frei wählbare Dateidatensätze werden verarbeitet.

Eine Direktzugriffsdateien ist eine Ansammlung von Datensätzen fester Lämge, von denen jeder aus 256 Bytes besteht. Den Datensätzen werden mit 1 beginnende Datensatznummern zugeordnet.

Direktzugriffsdateien werden für FDDs unterstützt. Eine Direktzugriffsdateien wird festgelegt, wenn der Parameter FOR INPUT bzw. FOR OUTPUT in der OPEN-Anweisung nicht gegeben wird.

Die Anweisungen GET und PUT werden für Ein- und Ausgabe für Direktzugriffsdateien verwendet. Durch diese Anweisungen erfolgt Eingabe bzw. Ausgabe eines bezeichneten Datensatzes über einen Ein-/Ausgabepuffer. Um Daten im Ein-/Ausgabepuffer zu manipulieren, wird die FIELD-Anweisung verwendet, um dem Ein-/Ausgabepuffer eine Zeichenvariable zuzuordnen. Um der einem Ein-/Ausgabepuffer zugeordneten Zeichenvariablen einen Wert zuzuordnen wird die LSET- oder die RSET-Anweisung anstatt der ASSIGN-Anweisung verwendet. Die ASSIGN-Anweisung kann zum Lesen von Daten aus einem Ein-/Ausgabepuffer verwendet werden.

Da eine dem Ein-/Ausgabepuffer zugeordnete Variable eine Zeichenvariable ist, sind Funktionen vorgesehen, um numerische Variablen in Zeichenvariablen umzuwandeln und umgekehrt. Diese Funktionen ermöglichen es, numerische Variablen direkt Zeichenvariablen zuzuschreiben, wodurch die Speicherwirksamkeit erhöht wird.

Wenn Direktverarbeitung durch die OPEN-Anweisung erklärt wird, so wird eine Datei mit dem bezeichneten Dateinamen festgelegt. Wenn keine Datei mit dem bezeichneten Namen gefunden wird, so wird eine neue Datei geschaffen.

Die LOP-Funktion wird verwendet, um die Größe der gegenwärtigen Datei anzuzeigen. Wenn in der PUT-Anweisung eine Datensatznummer gegeben wird, die die Dateigröße überschreitet, so wird die Datei automatisch erweitert.

Wenn in der GET- bzw. PUT-Anweisung keine Datensatznummer gegeben wird, so wird der Datensatz direkt nach dem vorher verarbeiteten Datensatz bezeichnet. Der erste Datensatz wird direkt nach Ausführung einer OPEN-Anweisung bezeichnet.

Die folgenden Anweisungen und Funktionen werden für Direktverarbeitung verwendet:

- Anweisungen: PUT, GET, FIELD, LSET, RSET
- Funktionen: LOC, LOF

Nachfolgend ist ein typisches Programm gezeigt:

MOUNT 2 ↵ (Diesen Befehl in Direktbetriebsart ausführen)

```
10 FIELD#1,10 AS NAME$,11 AS PRICE$,11 AS AMOUNT$
20 OPEN"TRANS" FOR INPUT AS #2
30 OPEN "GOODS" AS #1
40 IF EOF(1) THEN CLOSE:END
50 INPUT#2,CODE,QUANTITY:IF CODE<1 OR CODE>LOF(2)
   THEN PRINT "CODE ERROR":CLOSE:END
60 GET#1,CODE
70 RSET AMOUNT$=MKD$(QUANTITY*CVD(PRICE$)+CVD(PRICE$))
80 PUT#1,CODE
90 GOTO 40
```

9-3-1. FIELD-Befehl

```
FIELD [#] Dateinummer, Zeichenkettenlänge AS Zeichenvariablenname
          numerischer Ausdruck          einfacher Zeichenvariablenname
          [ , Zeichenkettenlänge AS Zeichenvariablenname ] *
          numerischer Ausdruck    einfacher Zeichenvariablenname
```

Funktion

Zuordnen einer Zeichenvariablen zu einem Ein-/Ausgabepuffer.

Parameter

(1)	Dateinummer	Ein als ganze Zahl bewerteter numerischer Ausdruck. $0 \leqslant$ Dateinummer < 16
(2)	Zeichenkettenlänge	Ein als ganze Zahl bewerteter numerischer Ausdruck. $0 \leqslant$ Zeichenkettenlänge < 256
(3)	Zeichenvariablenname	Ein einfacher Zeichenvariablenname.

Erklärung

(1) Dieser Befehl ordnet eine Variable einem durch eine Dateinummer bezeichneten Dateipuffer zu.

(2) Eine zuzuordnende Variable muß eine einfache Zeichenvariable sein. Numerische Variablen und Variablengruppenelemente können nicht bezeichnet werden.

(3) Die Länge der Zeichenkette darf 256 Bytes nicht überschreiten.

(4) Verwenden Sie die LET-Anweisung, um der dem Ein-/Ausgabepuffer zugeordneten Variablen einen Wert zu geben. Verwenden Sie für Zuweisung die Anweisungen LSET oder RSET.

(5) Dieser Befehl kann für nicht geöffnete Dateien verwendet werden.

(6) Die Dateinummer 0 bezeichnet den Systemein-/ausgabepuffer.

Beispiel

```
10 'FIELD SAMPLE
20 FIELD #1,128 AS H$,128 AS L$
30 OPEN "SAMPLE" AS #1
40 LSET H$="1st DATA OF SAMPLE"
50 LSET L$="2nd DATA OF SAMPLE"
60 PUT #1,1
70 CLOSE #1
80 END
```

9-3-2. GET-Anweisung

GET [#] Dateinummer [, Datensatznummer]

numerischer Ausdruck

Funktion

Lesen von Daten aus einem Ein-/Ausgabepuffer.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqq$ Dateinummer < 16

(2) Datensatznummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqq$ Datensatznummer < 65536

Erklärung

(1) Durch diese Anweisung wird der Inhalt des durch eine Datensatznummer bezeichneten Datensatzes in der durch eine Dateinummer bezeichneten Datei uber einen Ein-Ausgabepuffer gelesen.

(2) Diese Anweisung kann für eine für frei wählbare Verarbeitung geoffnete Datei ausgeführt werden.

(3) Wenn der Parameter "Datensatznummer" ausgelassen wird, so wird der Datensatz nach dem vorher verarbeiteten Datensatz bezeichnet. Der erste Datensatz wird direkt nach Ausführung einer OPEN-Anweisung bezeichnet.

9-3-3. PUT-Anweisung

PUT [#] Dateinummer [, Datensatznummer]

numerischer Ausdruck

Funktion

Schreiben von Daten von einem Ein-/Ausgabepuffer in eine Datei.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

(2) Datensatznummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Datensatznummer < 65536

Erklärung

(1) Durch diese Anweisung wird der Inhalt eines Ein-/Ausgabepuffers in einen durch eine Datensatznummer bezeichneten Datensatz in einer durch eine Dateinummer bezeichneten Datei ausgegeben.

(2) Diese Anweisung kann für eine für frei wählbare Verarbeitung geöffnete Datei ausgeführt werden.

(3) Wenn der Parameter "Datensatznummer" ausgelassen wird, so wird der Datensatz nach dem vorher verarbeiteten Datensatz bezeichnet. Der erste Datensatz wird direkt nach Ausführung einer OPEN-Anweisung bezeichnet.

Beispiel

```
 10 'GET&PUT SAMPLE
 20 FIELD #1,19 AS H$,19 AS L$
 30 '--PUT SAMPLE--
 40 OPEN "SAMPLE" AS#1
 50 FOR I=1 TO 10
 60 LSET H$="UPPER DATA("+STR$(I)+")"
 70 LSET L$="LOWER DATA("+STR$(I)+")"
 80 PUT #1,I
 90 NEXT I
100 CLOSE #1
110 '--GET SAMPLE--
120 OPEN "SAMPLE" AS #1
130 INPUT "LOCATION=";LP
140 IF LP>LOF(1) OR LP<1 THEN PRINT
    "NUMBER ERROR":CLOSE #1:END
150 GET #1,LP
160 PRINT H$
170 PRINT L$
180 PRINT:GOTO 130
```

```
RUN
LOCATION=4
UPPER DATA( 4)
LOWER DATA( 4)

LOCATION=7
UPPER DATA( 7)
LOWER DATA( 7)

LOCATION=11
NUMBER ERROR
```

9-3-4. RSET-Anweisung

RSET Zeichenvariablenname = Zeichenausdruck

Funktion

Eingabe des Wertes des Zeichenausdrucks auf der rechten Seite nach rechts ausgerichtet in die linke Variable.

Parameter

(1) Zeichenvariablenname: Ein einfacher Zeichenvariablenname.

(2) Zeichenausdruck

Erklärung

(1) Durch diese Anweisung wird der Wert des Ausdrucks auf der rechten Seite nach rechts ausgerichtet in die Zeichenvariable auf der linken Seite eingegeben.

(2) Es können nur die Zeichenvariablen verwendet werden, die den Ein-/Ausgabepuffern durch eine FIELD-Anweisung zugeordnet worden sind.

(3) Die Zeichenvariablen müssen einfache Zeichenvariablen sein: Variablengruppenelemente und numerische Variablen können nicht verwendet werden.

(4) Wenn der Wert des Zeichenausdrucks länger als die Länge der durch die FIELD-Anweisung bezeichneten Zeichenvariablen ist, so wird der überschüssige Teil auf der linken Seite abgeschnitten.

(5) Wenn der Wert des Zeichenausdrucks kürzer als die Länge der durch die FIELD-Anweisung bezeichneten Zeichenvariablen ist, so wird der linke Teil des Wertes mit Leerstellen gefüllt.

(6) Im Gegensatz zur LET-Anweisung gibt diese Anweisung nicht die Zuweisung einer Variablen zu einem Ein-/Ausgabepuffer frei.

Beispiel

```
 10 'RSET SAMPLE
 20 FIELD #1,15 AS H$,15 AS L$
 30 OPEN "SAMPLE" AS #1
 40 RSET H$="UPPER DATA"
 50 RSET L$="LOWER DATA"
 60 PUT #1,1
 70 CLOSE #1
 80 OPEN "SAMPLE" AS #1
 90 GET #1,1
100 PRINT "<";H$;">"
110 PRINT "<";L$;">"
120 CLOSE #1
130 END
```

```
RUN
<      UPPER DATA>
<      LOWER DATA>
```

9-3-5. LSET-Anweisung

LSET **Zeichenvariablenname = Zeichenausdruck**

Funktion

Eingabe des Wertes des Zeichenausdrucks auf der rechten Seite nach links ausgerichtet in die linke Variable.

Parameter

(1) Zeichenvariablenname: Ein einfacher Zeichenvariablenname.

(2) Zeichenausdruck

Erklärung

(1) Durch diese Anweisung wird der Wert des Ausdrucks auf der rechten Seite nach links ausgerichtet in die Zeichenvariable auf der linken Seite eingegeben.

(2) Es können nur die Zeichenvariablen verwendet werden, die den Ein-/Ausgabepuffern durch eine FIELD-Anweisung zugeordnet worden sind.

(3) Die Zeichenvariablen müssen einfache Zeichenvariablen sein: Variablengruppenelemente und numerische Variablen können nicht verwendet werden.

(4) Wenn der Wert des Zeichenausdrucks länger als die Länge der durch die FIELD-Anweisung bezeichneten Zeichenvariablen ist, so entfällt der überschüssige Teil.

(5) Wenn der Wert des Zeichenausdrucks kürzer als die Länge der durch die FIELD-Anweisung bezeichneten Zeichenvariablen ist, so wird der linke Teil des Wertes mit Leerstellen gefüllt.

(6) Im Gegensatz zur LET-Anweisung gibt diese Anweisung nicht die Zuweisung einer Variablen zu einem Ein-/Ausgabepuffer frei.

Beispiel

```
 10 'LSET SANPLE
 20 FIELD #1,15 AS H$,15 AS L$
 30 OPEN "SAMPLE" AS #1
 40 LSET H$="UPPER DATA"
 50 LSET L$="LOWER DATA"
 60 PUT #1,1
 70 CLOSE #1
 80 OPEN "SAMPLE" AS #1
 90 GET #1,1
100 PRINT "<";H$;">"
110 PRINT "<";L$;">"
120 CLOSE #1
130 END
```

```
RUN
<UPPER DATA     >
<LOWER DATA     >
```

9-3-6. MKSS/MKDS-Funktionen

MKS$ (Argument)

numerischer Ausdruck

MKD$ (Argument)

numerischer Ausdruck

Funktion

Dies sind Zeichenfunktionen, die den intern repräsentierten Wert eines Arguments als ihren Wert annehmen.

Parameter

(1) Argument — Ein numerischer Ausdruck, der automatisch in einen geforderten Typ umgewandelt wird, d.h.:

MKS$	Reeller Typ einfacher Genauigkeit
MKD$	Reeller Typ doppelter Genauigkeit

Erklärung

(1) Diese Funktionen nehmen den intern repräsentierten Wert eines Arguments als ihren Wert an.

(2) Die Länge des Wertes jeder Funktion ist wie folgt:

MKS$	6 Bytes
MKD$	11 Bytes

Beispiel

```
10 'MK~$ SAMPLE
20 FILED #1,6 AS S$,11 AS D$
30 OPEN "SAMPLE" AS #1
40 LSET S$=MKS$(SQR(2))
50 LSET D$=MKD$(SQR(2#))
60 PUT #1,1
70 CLOSE #1
80 ENDn PROG 3
```

9-3-7. CVS/CVD-Funktionen

CVS (Argument)

numerischer Ausdruck

CVD (Argument)

numerischer Ausdruck

Funktion

Diese Funktionen geben den von der durch den internen Kode eines Arguments repräsentierten Zeichenkette umgewandelten numerischen Wert.

Parameter

(1) Argument — Eine 255 Bytes nicht überschreitende Zeichenkette.

Erklärung

(1) Diese Funktionen nehmen den numerischen Wert an, dessen Wert von der Zeichenkette umgewandelt ist, die durch den internen Kode eines Arguments dargestellt wird.

(2) Typ und Länge des Arguments für jede Funktion sind wie folgt:

CVS	Reeller Typ einfacher Genauigkeit	6 Bytes
CVD	Reeller Typ doppelter Genauigkeit	11 Bytes

(3) Wenn die Länge des Arguments kürzer als die bezeichnete Länge ist, so wird die entsprechende Anzahl von Leerstellen (Kode 00) rechts vom Wert zugefügt.

(4) Wenn die Länge des Arguments länger als die bezeichnete Länge ist, so werden die überschüssigen Zeichen auf der rechten Seite des Wertes nicht beachtet.

Beispiel

```
 10 'CV■$ SAMPLE
 20 FIELD #1,6 AS S$,11 AS D$
 30 '--DATA OUT--
 40 OPEN "SAMPLE" AS #1
 50 LSET S$=MKS$(SQR(2))
 60 LSET D$=MKD$(SQR(2#))
 70 PUT #1,1
 80 CLOSE #1
 90 '--DATA IN--
100 OPEN "SAMPLE" AS #1
110 GET #1,1
120 PRINT "SINGLE=";CVS(S$)
130 PRINT "DOUBLE=";CVD(D$)
140 CLOSE #1
150 END
```

```
RUN
SINGLE= 1.141421
DOUBLE= 1.1414213562
73095
```

9-3-8. LOC-Funktion

LOC (Dateinummer)

numerischer Ausdruck

Funktion

Annahme der Datensatznummer nach dem letzten in frei wählbarer Verarbeitung verarbeiteten Satzes als Wert der Funktion.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Diese Funktion nimmt die Datensatznummer nach dem letzten in frei wählbarer Verarbeitung verarbeiteten Satzes als ihren Wert an.

(2) Verwendung dieser Funktion für eine nicht geöffnete Datei verursacht einen Fehler.

(3) Wenn eine OPEN-Anweisung ausgeführt wird, so nimmt diese Funktion den Wert 1 an.

Beispiel

```
 10 'LOC SAMPLE
 20 FIELD #1,20 AS H$
 30 '---DATA OUT---
 40 OPEN "SAMPLE" AS #1
 50 FOR I=1 TO 10
 60 LSET H$="DATA OF SAMPLE("+STR$(I)+")"
 70 PUT #1,I
 80 NEXT I
 90 CLOSE #1
100 '---DATA IN---
110 OPEN "SAMPLE" AS #1
120 FOR I=1 TO 10
130 SECPT=INT(RND(1)*10+1)
140 GET #1,SECPT
150 PRINT USING "(LOC=####)";LOC(1)
160 PRINT H$
170 NEXT I
180 CLOSE #1
190 END
```

```
RUN
(LOC=    7)
DATA OF SAMPLE( 6)
(LOC=    5)
DATA OF SAMPLE( 4)
(LOC=   10)
DATA OF SAMPLE( 9)
(LOC=    5)
DATA OF SAMPLE( 4)
(LOC=   11)
DATA OF SAMPLE( 10)
(LOC=    2)
DATA OF SAMPLE( 1)
(LOC=    2)
DATA OF SAMPLE( 1)
(LOC=   11)
DATA OF SAMPLE( 10)
(LOC=    5)
DATA OF SAMPLE( 4)
(LOC=    9)
DATA OF SAMPLE( 8)
```

9-3-9. LOF-Funktion

LOF (Dateinummer)

numerischer Ausdruck

Funktion

Annahme der Datensatzanzahl einer Datei als Wert der Funktion.

Parameter

(1) Dateinummer — Ein als ganze Zahl bewerteter numerischer Ausdruck.
$1 \leqslant$ Dateinummer < 16

Erklärung

(1) Diese Funktion nimmt die Datensatzanzahl einer durch die Dateinummer bezeichneten Datei als ihren Wert an. Der Wert ist ein ganzzahliger Typ.

(2) Verwendung dieser Funktion für eine nicht geöffnete Datei verursacht einen Fehler.

Beispiel

```
 10 'LOF SAMPLE
 20 FIELD #1,20 AS H$
 30 '---DATA OUT---
 40 PRINT " ---DATA OUT---"
 50 KILL "SAMPLE"
 60 OPEN "SAMPLE" AS #1
 70 FOR I=1 TO 10
 80 SEC=RND(1)*20+1
 90 PUT #1,SEC
100 PRINT USING "{SEC=#### LOF=####} ";SEC;LOF(1)
110 NEXT I
120 CLOSE #1
130 PRINT
140 '---DAATA IN---
150 PRINT "---DATA IN---
160 OPEN "SAMPLE" AS #1
170 SEC=1
180 IF SEC>LOF(1) THEN CLOSE:END
190 GET #1,SEC
200 PRINT USING "{SEC=#### LOF=####}";SEC;LOF(1)
210 SEC=SEC+1
220 GOTO 180
```

```
 ---DATA OUT---
{SEC=  12 LOF=  12}
{SEC=   7 LOF=  12}
{SEC=  19 LOF=  18}
{SEC=   7 LOF=  18}
{SEC=  21 LOF=  20}
{SEC=   1 LOF=  20}
{SEC=   3 LOF=  20}
{SEC=  20 LOF=  20}
{SEC=   7 LOF=  20}
{SEC=  15 LOF=  20}

---DATA IN---
{SEC=   1 LOF=  20}
{SEC=   2 LOF=  20}
{SEC=   3 LOF=  20}
{SEC=   4 LOF=  20}
{SEC=   5 LOF=  20}
{SEC=   6 LOF=  20}
{SEC=   7 LOF=  20}
{SEC=   8 LOF=  20}
{SEC=   9 LOF=  20}
{SEC=  10 LOF=  20}
{SEC=  11 LOF=  20}
{SEC=  12 LOF=  20}
{SEC=  13 LOF=  20}
{SEC=  14 LOF=  20}
{SEC=  15 LOF=  20}
{SEC=  16 LOF=  20}
{SEC=  17 LOF=  20}
{SEC=  18 LOF=  20}
{SEC=  19 LOF=  20}
{SEC=  20 LOF=  20}
```

9-4. Dateihandhabung

9-4-1. FORMAT-Befehl

FORMAT [Dateideskriptor]
Anzahl der Zeichen

Funktion

Initialisieren der Diskette für FDD (Floppy Disk Laufwerk).

Parameter

(1) Dateideskriptor

Erklärung

(1) Dieses Befehl initialisiert Die Diskette für ein FDD.

(2) Der Gerätname im Dateideskriptor muß "0" sein.

(3) Der Dateideskriptor kann ausgelassen werden.

(4) Lassen Sie bei der Durchführung dieses Befehls angemessene Sorgfalt walten, da alle Dateien auf dem Medium durch die Initialisierung gelöscht werden.

(5) Ausführung dieses Befehls für eine Diskette mit einer offenen Datei verursacht einen Fehler.

9-4-2. FILES-Befehl

FILES [Dateideskriptor]

Zeichenausdruck

Funktion

Anzeige des Attributs einer Datei eines FDD.

Parameter

(1) Dateideskriptor Ein Zeichenausdruck

Erklärung

(1) Durch diesen Befehl wird das Attribut einer Datei eines FDD angezeigt.

(2) Nur der Gerätname des Dateideskriptors "0" ist gültig.

(3) Der Parameter "Dateideskriptor" kann ausgelassen werden.

Beispiel

```
FILES
ABCDEFGH.123  S  B  P  P
     10
SAMPLE   .    S  A  P
      2
DATA     .CK1 S  A  D
      7
DATA     .CK2 S  A  D
      7
TEST         F
      3
```

9-4-3. KILL-Befehl

KILL **Dateideskriptor**

Zeichenausdruck

Funktion

Löschen einer bezeichneten Datei auf dem FDD.

Parameter

(1) Dateideskriptor

Erklärung

(1) Durch diesen Befehl wird die Datei auf dem FDD gelöscht.

(2) Wenn die bezeichnete Datei nicht gefunden wird, so wird hierdurch kein Fehler verursacht (die Durchführung wird fortgesetzt).

(3) Eine geöffnete Datei kann nicht gelöscht werden.

(4) Wenn im Dateideskriptor kein Gerätname bezeichnet ist, so wird FDD 0 bezeichnet.

Beispiel

```
KILL "SAMPLE"
```

9-5. Programmdateien

Programme können als Dateien gehandhabt werden. Vom Standpunkt ihres Darstellungsformats aus können Programme in interne Formatprogramme und ASCII-Formatprogramme klassifiziert werden. Das interne Format ist ein im Hauptspeicher verwendetes kondensiertes Darstellungsformat. Das ASCII-Format ist ein normales Darstellungsformat, wie es in der LIST-Anweisung usw. gesehen werden kann.

SAVE Dateideskriptor [, Kennwort] [, A]

Zeichenausdruck

Funktion

Aufzeichnen eines Programms in einer bezeichneten Datei.

Parameter

(1)	Dateideskriptor	Ein Zeichenausdruck.
(2)	Kennwort	Eine Zeichenkonstante von 1 bis 8 Zeichen, eingeschlossen in Anführungszeichen (" ").
(3)	A	Bezeichnung des ASCII-Formats. Wenn dieser Parameter ausgelassen wird, so wird das interne Format bezeichnet.

Erklärung

(1) Durch diesen Befehl wird der Inhalt des gegenwärtig bezeichneten Programmbereichs zu der durch den Dateideskriptor bezeichneten Datei ausgegeben.

(2) Das Kennwort ist eine von " " eingeschlossene 1 bis 8 Zeichen lange Zeichenkonstante.

(3) Wenn der Parameter "A" ausgelassen wird, so wird der Inhalt in internem Format ausgegeben. Das interne Format ist das im Hauptspeicher verwendete Format zur Programmdarstellung.

(4) Wenn der Parameter "A" bezeichnet wird, so wird der Inhalt für die Ausgabe in das ASCII-Format umgewandelt. Das ASCII-Format ist ein Format der Darstellung durch alphanumerische Zeichen, wie es durch eine LIST-Anweisung gesehen werden kann.

(5) Dieser Befehl kann nicht für offene Dateien durchgeführt werden.

Wenn ein Kennwort für den gegenwärtig bezeichneten Programmbereich bezeichnet worden ist, so wird es zusammen mit dem Inhalt des Programmbereichs ausgegeben.

(6) Wenn ein Kennwort für alle Programmbereiche bezeichnet worden ist, so kann dieser Befehl nicht ausgeführt werden.

(7) Wenn ein Kennwort bezeichnet worden ist, so kann dieser Befehl nicht in Direktbetriebsart ausgeführt werden. Wenn ein Kennwort in einem Programm geschrieben ist, so kann dieser Befehl ausgeführt werden. Wenn die SAVE-Anweisung kein Kennwort enthält, so wird das Computerkennwort ausgegeben. Wenn die SAVE-Anweisung ein Kennwort enthält, so wird dies ausgegeben.

Beispiel

```
SAVE "CAS0: TEST.1", "PASSWORD", A
```

9-5-2. LOAD-Befehl

LOAD Dateideskriptor [, R]

Zeichenausdruck

Funktion

Laden eines Programms in den Speicher.

Parameter

(1) Dateideskriptor — Ein Zeichenausdruck.

(2) R — Bezeichnet daß das Programm nach dem Laden sofort auszuführen ist.

Erklärung

(1) Durch diesen Befehl wird ein Programm von einer durch den Dateideskriptor bezeichneten Datei in den gegenwärtigen Programmbereich geladen. Die Datei kann internes Format oder ASCII-Format haben.

(2) Wenn vor der Ausführung dieses Befehls ein Programm im gegenwärtigen Programmbereich vorhanden ist, so wird dieses Programm gelöscht.

(3) Wenn der Parameter "R" nicht bezeichnet wird, so schließt das System eine offene Datei und wartet auf weitere Befehle.

(4) Wenn der Parameter "R" bezeichnet wird, so wird das Programm vom Anfang an ausgeführt. In diesem Fall wird eine offene Datei nicht geschlossen.

(5) Wenn ein Kennwort für alle Programmbereiche oder für den gegenwärtigen Programmbereich bezeichnet worden ist, so kann dieser Befehl nicht in direkter Betriebsart ausgeführt werden.

(6) Wenn ein Kennwort für das zu ladende Programm bezeichnet worden ist, so wird dieses Kennwort für den gegenwärtigen Programmbereich bezeichnet.

(7) Wenn LOAD "COM0:" durch einen anderen Computer durch Verwendung der Datenübertragungsleitung ausgeführt wird, so muß das zu sendende Programm zuerst im ASCII-Format gespeichert werden und dann als sequentielle Datei ausgegeben werden.

Beispiel

Übertragung vom FP-1100 zum FP-200.

Sender (FP-1100)

```
10 FOR I=1 TO 10
20 PRINT SQR(CDBL(I#)),
30 NEXT I
```

SAVE "SQR", A ↵

NEW ↵ MOUNT 2 ↵

```
10 OPEN "SQR" FOR INPUT AS #1
20 OPEN"COM0:" FOR OUTPUT AS #2
30 IF EOF(1)=-1 THEN CLOSE:END
40 INPUT #1,A$:PRINT A$
50 PRINT #2,A$
60 GOTO 30
```

Empfänger (FP-200)

LOAD "COM0:" ↵

Nochmals der Sender

RUN ↵

(8) LOAD ', R' initiiert ein Zufallszahlenfeld, den DEF-Typ, die vom Benutzer definierten Funktionen und Datenzeiger.

10. KASSETTENTONBANDGERAT (CAS 0:)

10-1. Dateideskriptor

CAS 0 : Dateiname

Die Übertragungsrate (Baudrate) für ein Kassettentonbandgerät ist 300 Baud.

10-2. Umriß der Verarbeitung

Das Kassettentonbandgerat wird als Eingabedatei oder als Ausgabedatei fur sequentielle Verarbeittung verwendet. Aus diesem Grund konnen die folgenden Anweisungen fur das Kassettentonbandgerät verwendet werden:

- OPEN (FOR INPUT/FOR OUTPUT)
- PRINT #, PRINT # USING
- INPUT #
- CLOSE

Für Ein- und Ausgabe von Programmen können die folgenden Befehle verwendet werden:

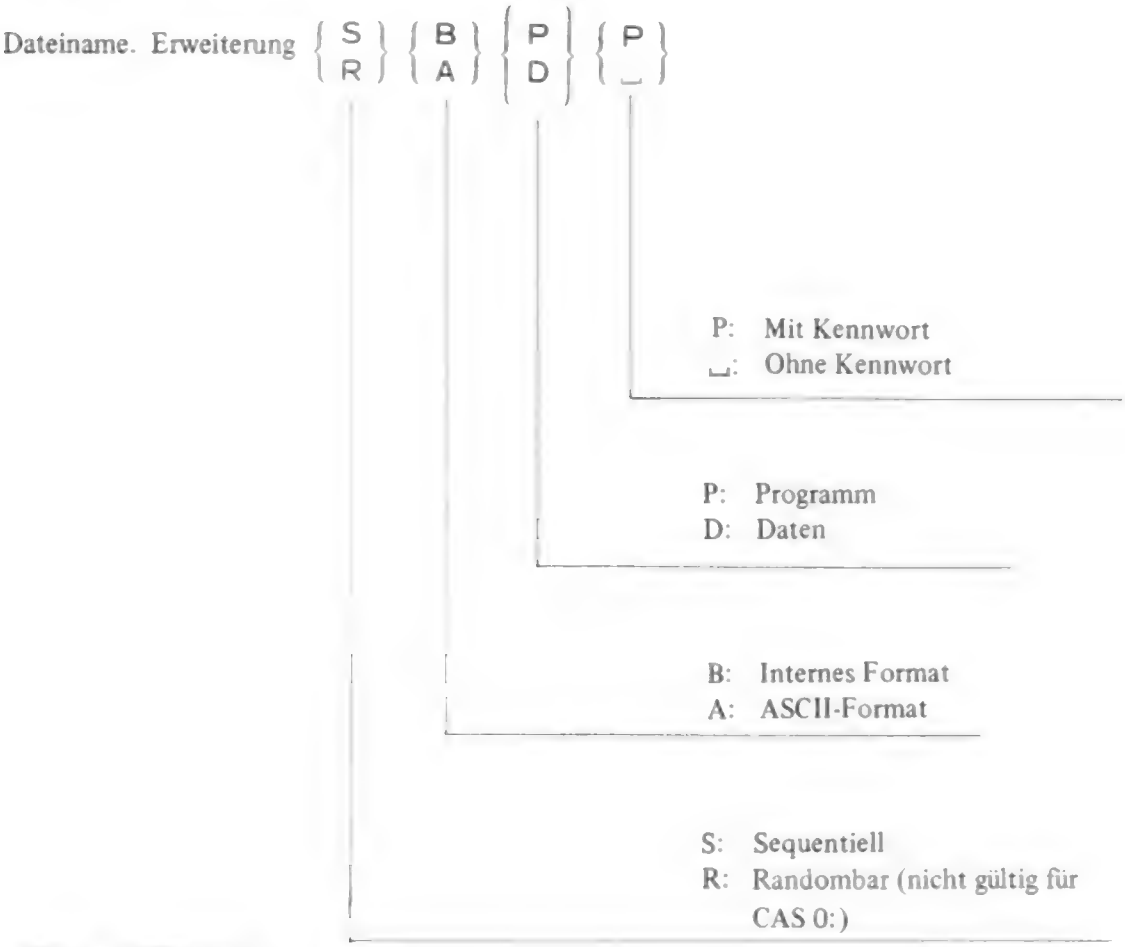
- LOAD, SAVE,

Zusätzlich sind Spezialbefehle fur das Kassettentonbandgerät (CAS 0:) erhältlich:

- VERIFY

Eine Kassettentonbanddatei wird durch ihren Dateinamen verwaltet. Sie kann ein Kennwort haben. Es kann jeweils nur eine Datei zur Zeit gehandhabt werden. Wenn Dateien durch eine OPEN-Anweisung (FOR INPUT) usw. gesucht werden, so werden die während des Suchbetriebs gefundenen Dateinamen und Attribute der Dateien angezeigt. Die Attribute werden wie folgt angezeigt:

Für eine BASIC-Datei:



Für eine CETL-Datei:

Beispiel

```
LOAD"CASO:DUMPMEM.BA
S"
TEST     .BAS S B P
SAMP     .BAS S B P P
SAMPLE   .     S A D
editfile.      S A D
NAMETABL.BAS S A D
FILEBIRD.      F
DUMPMEM .BAS S B P
```

10-3. VERIFY-Befehl

VERIFY **Dateideskriptor**
Zeichenausdruck

Funktion

Uberprüfen von Dateien auf einem Kassettentonbandgerät.

Parameter

(1) Dateideskriptor Ein Zeichenausdruck.

Erklärung

(1) Durch diesen Befehl wird eine durch einen Dateideskriptor bezeichnete Datei auf einem Kassettentonband überprüft.

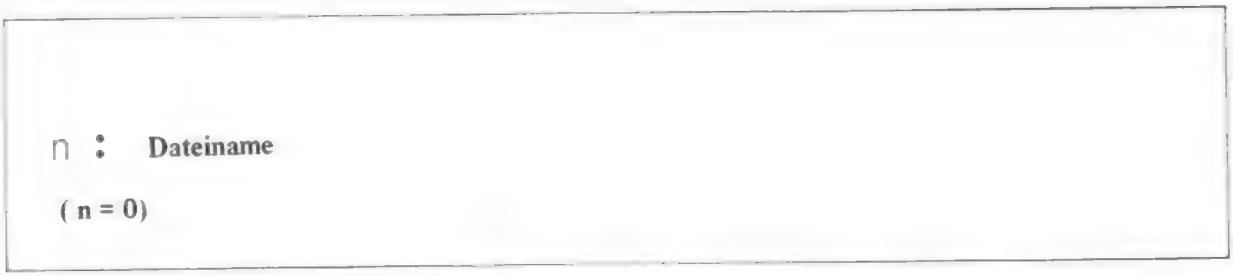
(2) Die Prüfung wird unter Verwendung von Parität und Prüfsumme durchgeführt, die in der zu überprüfenden Datei enthalten sind.

Beispiel

VERIFY "CAS0: STAR"

11. FDD

11-1. Dateideskriptor



11-2. Umriß der Verarbeitung

FDDs können für sequentielle und für Direktverarbeitung verwendet werden. Aus diesem Grund können alle für Dateiverarbeitung zur Verfugung stehenden Anweisungen und Befehle für FDDs verwendet werden.

Die Dateien werden durch ihre Dateinamen verwaltet. Für die FDD-Dateien kann ein Kennwort bezeichnet werden. Die Dateiattribute können durch Verwendung des FILES-Befehls überprüft werden.

Für eine BASIC-Datei:

Dateiname. Erweiterung { S / R } { B / A } { P / D } { P / ␣ } n

Kapazität: Anzahl der Blöcke (in 256 Bytes)

P: Mit Kennwort
␣: Ohne Kennwort

P: Programm
D: Daten

B: Internes Format
A: ASCII-Format

S: Sequentiell
R: Randombar (nicht gültig für CAS 0:)

Für eine CETL-Datei:

Dateiname F n

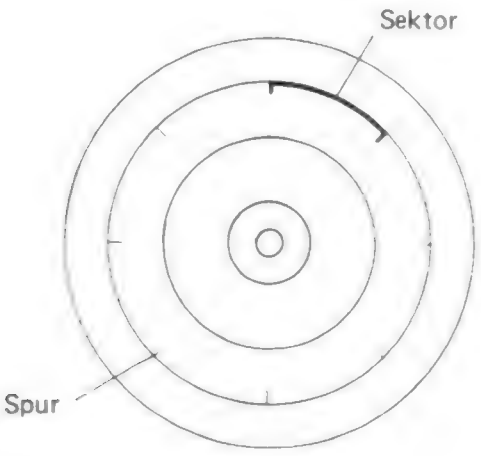
Kapazität: Anzahl der Blöcke (in 256 Bytes)

F: Sequentiell

Beispiel

```
SAMPLE   .     S A P P
  2
DATA     .     S A D
    15
TEST     .BAS  S B P
  7
FILEBIRD       F
  5
```

Bis zu 33 Dateien konnen auf einer Diskette gespeichert werden. Die Dateidaten werden in Anhäufungen (2.048 Bytes) verwaltet. Jede Anhäufung besteht aus acht Sektoren, und jeder Sektor - die physikalische Verarbeitungseinheit - besteht aus 256 Bytes. Die maximale Dateikapazität pro Diskette ist 33 Anhäufungen (264 Sektoren bzw. 67.584 Bytes). Zusätzlich sind zwei Anhäufungen (16 Sektoren bzw. 4.096 Bytes) als Verwaltungsbereich reserviert.
Eine Diskette ermoglicht Aufzeichnung von Daten auf einer Seite. Die Seite hat 35 Spuren (0 bis 34) in konzentrischen Kreisen. Jede Spur hat 8 Sektoren (1 bis 8). Der Sektor ist die kleinste physikalische Einheit für Lese-und Schreibbetrieb. Jeder Sektor besteht aus 256 Bytes. Für die erste Verwendun muß die Diskette durch den FORMAT-Befehl initialisiert werden.



12. DATENLEITUNGEN

12-1. Dateideskriptor

```
COM0:
```

Die Übertragungsrate für die Datenübertragungsleitung ist 300 Baud.

12-2. Umriß der Verarbeitung

COM-Dateien können nur für sequentielle Verarbeitung verwendet werden. Sie können als Eingabedateien, Ausgabedateien oder als Ein-/Ausgabedateien bezeichnet werden.

```
OPEN "COM 0: – "FOR INPUT AS –      Eingabedatei
OPEN "COM 0: – "FOR OUTPUT AS –     Ausgabedatei
```

Datenaustausch mit externen Geräten erfolgt über eine Datenübertragungsleitung. Diese Datenübertragungsleitung ist über einen RS-232C-Anschluß an das externe Gerät angeschlossen. Daten werden mit Start-Stop-Synchronisierung (asynchron) übertragen.

Die Eingabedaten für jeden Kanal werden einmal in einem Pufferbereich von 256 Bytes gespeichert und dann durch Verwendung der INPUT=-Anweisung usw. abgerufen. Dieser Puffer verhütet Fehlen von Eingabedaten.

Es ist auch möglich, ein Programm vorzubereiten, das sofortige Verarbeitung von Eingabedaten durch die Datenleitungsunterbrechungsfunktion ermöglicht.

13. CETL

13-1. Konzepte für CETL

CETL ist ein Programm, das Berechnungen, Redigieren und Sortieren einfach durch Eingabe von Zahlen, Zeichen oder Ausdrücken in eine einzige festgelegte Tabelle erleichtert und so Verwendung durch einen unerfahrenen Benutzer erlaubt.

13-1-1. Eigenschaften von CETL

1. Einfaches und leichtverständliches Befehlssystem.
   Es sind nur sechzehn Befehle vorhanden. Jeder Befehl besteht aus dem Anfangsbuchstaben des Wortes, das den Befehl beschreibt, wodurch die Befehle einfach behalten werden können.
2. Eine Datei kann in 10 Segmente unterteilt werden. Betrieb zwischen Dateien ist auch möglich.
   Es können bis zu zehn unabhängige Dateien erzeugt werden, so daß verschiedene Aufgaben ohne externe Speicherung möglich sind. Zusätzlich kann die FL-Funktion verwendet werden, um Einschub in eine andere Datei oder Bezug auf eine andere Datei durchzuführen.
3. Verknüpfung von CETL und BASIC
   In BASIC geschriebene Programme oder Operationsausdrücke können von CETL abgerufen werden, und Daten in CETL können durch BASIC verarbeitet werden, wodurch leistungsfähige Datenverarbeitung möglich wird.

13-1-2. Funktionen von CETL

CETL kann die folgenden Aufgaben sehr einfach und schnell einfach durch aufeinanderfolgende Eingabe von Daten in eine Tabelle durchführen.

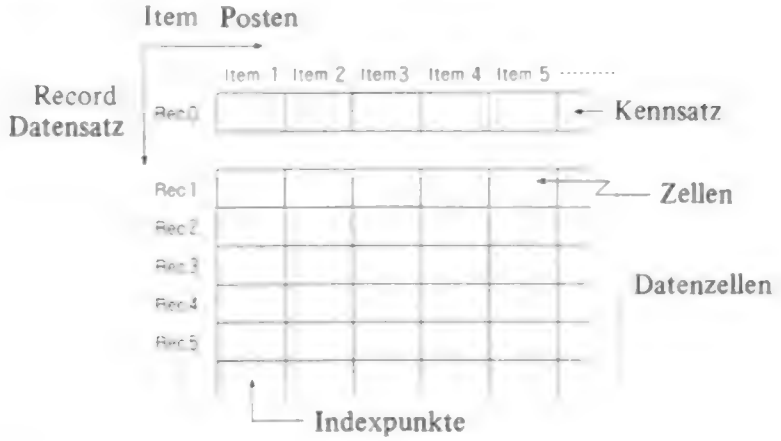
(1) Redigieren von Daten
(2) Berechnungen
(3) Sortieren
(4) Widerauffinden
(5) Erstellen von Tabellen

13-2. Dateiorganisation

Die durch CETL erzeugte Datei besteht aus:

(1) Dateiname
(2) Kennsatz
(3) Datenzellen (tatsächlicher Datenbereich)

Dateiname



- Dateiname
  Dies ist der Name der Datei, der eine Länge von bis zu acht Zeichen haben kann. Beachten Sie bitte, daß die folgenden Zeichen nicht verwendet werden können: "(22H), , (2CH), : (3AH), * (2EH)

- Posten
  Dies ist der Name einer Spalte in einer Datentabelle.

- Datensatz
  Dies ist der Name einer Zeile in einer Datentabelle.

- Kennsatz
  Dies ist der Bereich, der den Inhalt jedes Postens anzeigt, und er besteht aus den folgenden Elementen:

Postenname	:	Der Name des Postens.
Typ	:	Anzeige, ob die im Posten gespeicherten Daten Zeichendaten oder Zifferndaten sind.
Ausdruck	:	Bezeichnung des Ausdrucks, der erforderlichenfalls den im numerischen Datenwort zu speichernden Wert angibt.
Format	:	Bezeichnung des Formats für Datenausgabe.

- Zelle
  Dies ist der Bereich, in dem Daten gespeichert werden, und die Zelle ist das kleinste Element einer Datentabelle.

13-3. Für CETL anwendbare Befehle

In CETL-Betriebsart stehen sechzehn CETL-Befehle und siebzehn BASIC-Befehle zur Verfügung.

BASIC-Befehle:
SYSTEM, CLEAR, MOUNT, PRINT, LPRINT, PASS, KEY, RANDOMIZE, LET, ANGLE, FILES, FORMAT, KILL, VERIFY, AREA, RESET, FILE.
Die obigen Befehle können wie in BASIC-Betriebsart verwendet werden, aber Mehrfachanweisungszeilen sind nicht zulässig.

CETL-Befehle

Redigieren	A (automatische Eingabe), D (Datei loschen), I (Datei einschieben), K (Abbruch), M (Datei bewegen), N (Neue Datei), R (neu benennen)
Betrieb	B (Leerstelle), C (Berechnung), F (Finden), J (Sprung), L (Auflisten), S (Sortieren), T (Tabelle)
Ein-/Ausgabe	G (Lesen), P (Schreiben)

13-4. Eingabebereich für Datensatz und Posten

Zahlen	Datensatz 1 bis 65535 Posten 1 bis 255
Eingabeformat	
Numerischer Ausdruck:	Numerischer Ausdruck, Funktion, numerischer Wert, Variable
Zeichenausdruck:	Zeichenausdruck, Funktion, Zeichenvariable

13-5. PF-Tasten in CETL-Betriebsart

Der Betrieb der PF-Tasten in der CETL-Betriebsart unterscheidet sich von dem in der BASIC-Betriebsart und bewirkt Ausführung oder Anzeige verschiedener Befehle oder CETL-Verwaltungsfunktionen.

(1) Taste PF0

- Umschalten zwischen Redigierbetrieb und Befehlsbetrieb.
- Wenn diese Taste wahrend Befehlsbetriebsart (Warten auf Befehlseingabe) gedruckt wird, so wird der Inhalt der gegenwartigen Datenzelle angezeigt, der Positionsanzeiger "_" bewegt sich zum Ende der Daten, und es erfolgt Eintritt in die Redigierbetriebsart.
- Wenn die Taste **PF0** gedrückt wird und Eintritt in die Redigierbetriebsart erfolgt, während die anzuzeigenden Daten mehr als 160 Zeichen enthalten, so werden nur die ersten 160 Daten angezeigt, und der Positionsanzeiger bewegt sich zur rechten unteren Ecke der Anzeige.

(2) Taste PF1

- Anzeige von "FILE" (Date). Geben Sie einen Wert von 0 bis 9 ein und drücken Sie die Taste ↵ . Die Anzeige wechselt dann zu dem Dateibereich, der dem eingegebenen Wert entspricht.

(3) Taste PF2

- Diese Taste hat die gleiche Funktion wie SYSTEM ↵ , und die Große des verwendeten CETL-Bereichs und jedes Dateibereichs wird in Bytes angezeigt.

(4) Taste PF3

- Der gesamte Bildschirm wird geloscht, und die Liste der CETL-Befehle wird angezeigt.
- Durch Druck auf die Taste STOP/CONT oder die Taste BREAK kann die Anzeige angehalten bzw. unterbrochen werden.

(5) Taste PF4

- Diese Taste hat die gleiche Funktion wie PRINT FRE ↵ , und sie zeigt die Große des nicht verwendeten CETL-Bereichs an.

(6) Taste PF5 (SHIFT + PF0)

- Diese Taste hat die gleiche Funktion wie **CTRL + N** , und sie schaltet die Tastatur zu NUM-Betriebsart um. In NUM-Betriebsart konnen einige alphanumerische Tasten und Symboltasten als Hilfszahlentasten verwendet werden. (Siehe Seite 233.)
- Wenn diese Taste wahrend der NUM-Betriebsart gedruckt wird, so wird die NUM-Betriebsart aufgehoben (Umschalter).

(7) Taste PF6 (SHIFT + PF1)

- Anzeige von "FL(". Diese Taste wird verwendet, wenn die FL-Funktion verwendet werden soll.

(8) Taste PF7 (SHIFT + PF2)

- Anzeige von "RC(". Diese Taste wird verwendet, wenn die RC-Funktion verwendet werden soll.

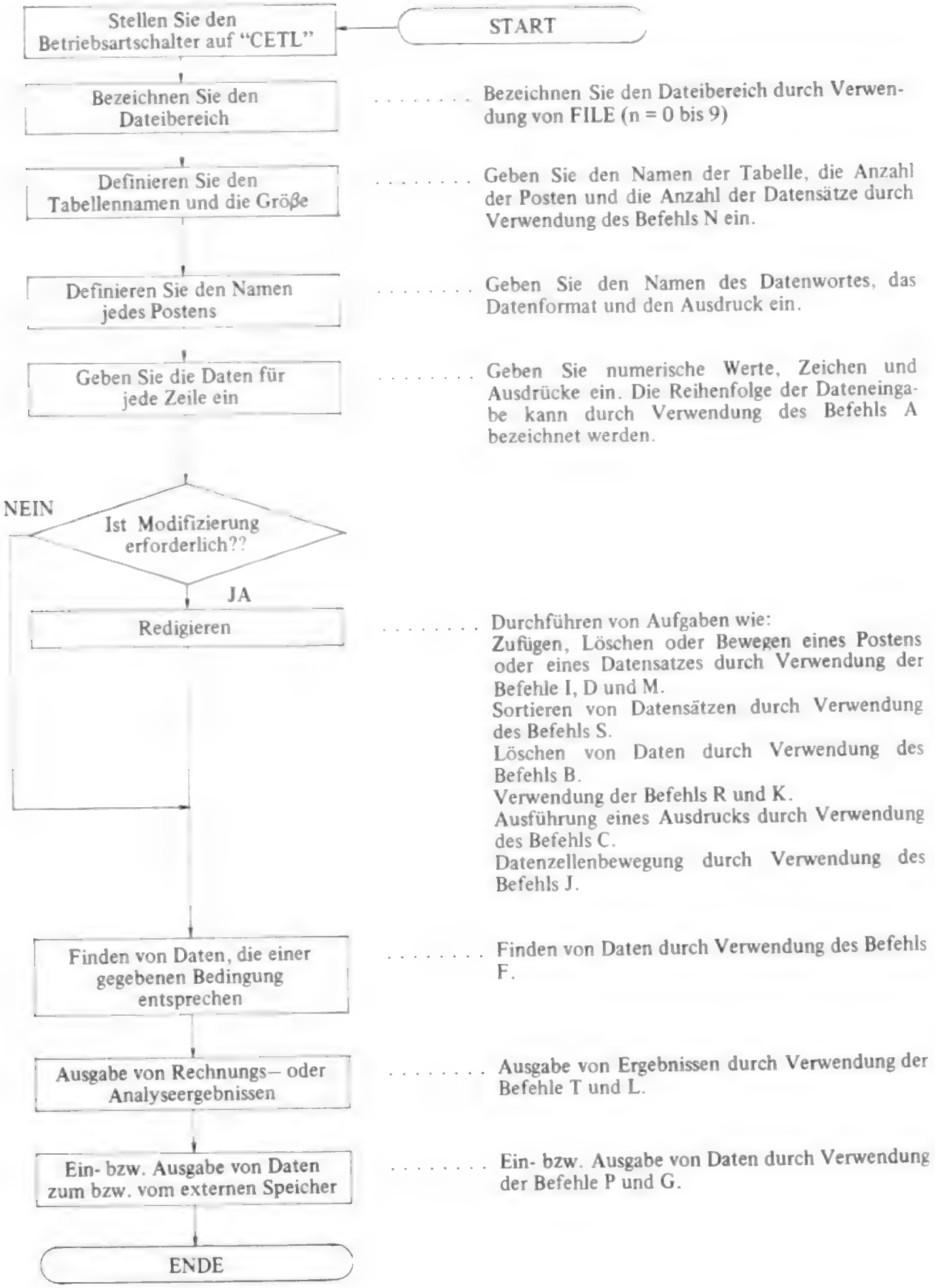
(9) Taste PF8 (SHIFT + PF3)

- Anzeige von "IT (". Diese Taste wird verwendet, wenn die IT-Funktion verwendet werden soll.

(10) Taste PF9 (SHIFT + PF4)

- Diese Taste hat die gleiche Funktion wie PRINT DATE$, TIME$, ↵ und zeigt Datum und Zeit an.

13-6. CETL-Betriebsverfahren



13-7. CETL-Befehle

13-7-1. N (neue Datei)

```
N
```

Funktion

Erstellen einer neuen Datei

Erklärungen

(1) Durch diesen Befehl werden der Dateiname, die Anzahl der Posten und die Anzahl der Datensätze eingegeben.

(2) Wenn dieser Befehl ausgeführt wird, während schon eine Datei in dem Dateibereich existiert, so tritt ein Fehler ein. Ändern Sie deshalb den Dateibereich durch Verwendung von "FILE n", oder löschen Sie die Datei durch Ausführung des Befehls K vor der Ausführung dieses Befehls.

(3) Geben Sie den Dateinamen, die Anzahl der Datensätze und die Anzahl der Posten nach dem Befehl ein.

Dateiname: Bis zu 8 Zeichen

Anzahl der Datensätze: Bis zu 65535
Anzahl der Datenworte: Bis zu 255
} Abhängig von der Speicherkapazität

* Speicherkapazität = Anzahl der Datensätze x Anzahl der Posten + Anzahl der Datensätze x 2 + Anzahl der Posten x 4 + Dateinamenlänge + 4 + Gesamtanzahl der in jeder Zelle geschriebenen Zeichen.

(4) Geben Sie nach der Eingabe des Dateinamens, der Anzahl der Datensätze und der Anzahl del Posten den Namen jedes Postens, den Typ, den Ausdruck und das Ausgabeformat ein, und beginnen Sie dann mit der Dateneingabe.

Beispiel

```
>N

File Name?TEST          ← Geben Sie den Dateinamen ein.
Number of Rec.?15       ← Geben Sie die Anzahl der Datensätze ein.

Number of Item?6        ← Geben Sie die Anzahl der Posten ein.
Item1:Name?_            ← Beginnen Sie dann mit Eingabe von
                          Postenname usw.
```

A

Funktion

Einstellung der Reihenfolge für Dateneingabe.

Erklärungen

(1) Nach Eingabe des Befehls A wird eine Nachricht angezeigt, um zu fragen ob die Daten in Datensatzreihenfolge (horizontal) oder in Datenwortreihenfolge (vertikal) eingegeben werden. Geben Sie "R" für Datensatzreihenfolge und "I" für Datenwortreihenfolge ein.

(2) Wenn eine neue Datei mit dem Befehl N erstellt wird, so geben Sie zuerst den Dateinamen, die Anzahl der Datensätze und die Anzahl der Posten ein; schalten Sie dann durch Druck auf die Taste **PF0** auf Befehlsbetriebsart um und führen Sie den Befehl A aus.

(3) Nach der Ausführung dieses Befehls wartet das System auf den nächsten Befehl. Drücken Sie die Taste **PF0** um Redigierbetriebsart zu beginnen, oder verwenden Sie den Befehl J, um für Dateneingabe zu der festgelegten Zelle zu gehen.

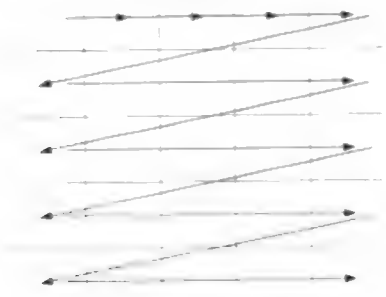
(4) Dieser Befehl ist Zusätzlich zur Eingabe neuer Daten auch Nützlich zum Ändern von Daten in einem Posten.

Beispiel

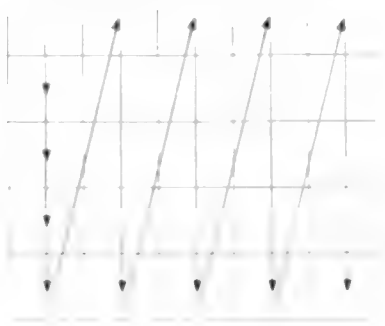
```
>A
Rec/Item(R/I)?I
Ready F0
>
```

Bezeichnen Sie die Reihenfolge der Posten

R → AUTO-R-Betriebsart



I → AUTO-I-Betriebsart



13-7-3. I (Datei einschieben)

```
I
```

Funktion

Eingabe eines zusätzlichen Datensatzes oder Datenpunkts.

Erklärungen

(1) Durch diesen Befehl wird ein Datensatz oder ein Datenpunkt eingeschoben, wenn zusätzliche Daten nach der Datenersteingabe einzugeben sind.

(2) Legen Sie nach Eingabe des Befehls I fest, ob ein Datensatz oder ein Posten einzugeben ist und wo die Daten einzuschieben sind. (Die Daten werden an der Position nach dem bezeichneten Datensatz bzw. Posten eingeschoben. Wenn Daten als Startpunkt zuzufügen sind, so geben Sie die Position als 0 an.)

(3) Nach Angabe der Position können zusätzliche Daten eingegeben werden.

(4) Nach der Ausführung dieses Befehls erfolgt Neunumerierung für die Datensatznummern bzw. Datenwortnummern.

Beispiel

```
>I

Rec/Item(R/I)?R
```
Bezeichnung des Datensatzes.

```
Rec.?4
5-1 ?_
```
Einschieben nach Datensatz 4.

```
>I

Rec/Item(R/I)?I
```
Bezeichnung des Datenwortes.

```
Item?2
Item3:Name?_
```
Einschieben nach Datenwort 2.

13-7-4. D (Datei löschen)

```
D
```

Funktion

Löschen mehrerer Datensätze oder Datenworte.

Erklärungen

(1) Durch diesen Befehl werden eventuell nicht erforderliche Datensätze oder Posten nach der Dateneingabe gelöscht.

(2) Bezeichnen Sie nach Eingabe des Befehls D den Datensatz bzw. den Posten und die zu löschenden Daten.

(3) Der Bereich für die zu löschenden Posten muß innerhalb der gegenwaärtig existierenden Datensatz- bzw. Postennummern liegen. Wenn Datensätze oder Posten bezeichnet werden, die nicht existieren, so wird ein Fehler verursacht. Beachten Sie bitte, daß die Datensatznummer 0 nicht bezeichnet werden kann.

(4) Nach der Ausführung dieses Befehls werden die Datensatznummern (bzw. die Postennummern) neu nummeriert.

Beispiel

```
>D
Rec/Item(R/I)?I          ← Bezeichnung eines Postens.
Item?8                   ← Löschen von Datenwort 8.
Ready F0
>_
```

```
>D
Rec/Item(R/I)?R          ← Bezeichnung eines Datensatzes.
Rec.?10,12               ← Löschen der Datensätze 10 bis 12.
Ready F0
>_
```

13-7-5. M (Datei bewegen)

```
M
```

Funktion

Bewegen mehrerer Datensätze oder Posten.

Erklärungen

(1) Durch diesen Befehl werden die bezeichneten Daten in Einheiten von gesamten Datensätzen oder Posten bewegt.

(2) Bezeichnen Sie nach Eingabe des Befehls M den Datensatz bzw. den Posten und den Bereich der zu bewegenden Daten.

(3) Der Bereich für die Bewegung und das Ziel der Bewegung müssen im Bereich der gegenwärtig existierenden Datensätze bzw. Posten liegen, da sonst ein Fehler verursacht wird.

(4) Nach der Ausführung des Befehls M werden die Datensatznummern bzw. die Postennummern neu nummeriert.

(5) Wenn das Ziel der Bewegung innerhalb des zu bewegenden Datenbereichs liegt, so wird ein Fehler verursacht.

Beispiel

```
>M
Rec/Item(R/I)?I
Location flom?2,3
Location to?5
Ready F0
>_
```

Bezeichnung eines Postens.

Bewegung von Posten 2 zu Posten 3.

Bewegung zur Position nach Posten 5.

```
>M
Rec/Item(R/I)?R
Location flom?5
Location to?11
Ready F0
>_
```

Bezeichnung eines Datensatzes.

Bewegen von Datensatz 5.

Bewegung zur Position nach Datensatz 11.

13-7-6. R (Neu benennen)

```
R
```

Funktion

Neubenennung einer Datei.

Erklärungen

(1) Änderung des gegenwärtigen Dateinamens zu einem anderen Namen.

(2) Wenn der Befehl R eingegeben wird, so wird der gegenwärtige Dateiname angezeigt, und der neue Dateiname wird angefordert.

Geben Sie den neuen Dateinamen ein (bis zu acht Zeichen).

* Die gultigen Zeichen für Dateinamen sind Zeichen mit den Koden 20H bis 7FH und 80H bis FEH, ausgeschlossen "(Kode 22H), * (Kode 2EH), : (Kode 3AH) und , (Kode 2CH).

Beispiel

```
>R
Old Name:SALES
New Name?PROCEEDS
Ready F0
>_
```

Gegenwärtiger Dateiname

Neuer Dateiname

13-7-7. K (Abbruch)

```
K
```

Funktion

Löschen von Dateien.

Erklärungen

(1) Dieser Befehl loscht eine Datei im gegenwärtigen Dateibereich oder alle Dateien im CETL-Dateibereich.

(2) Bezeichnen Sie nach der Eingabe des Befehls K entweder "alle Dateien" (A) oder nur die gegenwärtig bezeichnete Datei (P) für Löschen.

(3) Achten Sie darauf, diesen Befehl nicht versehentlich auszuführen, da hierdurch Dateien geloscht werden. Wenn dieser Befehl versehentlich ausgeführt wurde, so drücken Sie die Taste **BREAK** nach der Anzeige von A/P ? und vor Druck auf die Taste ↵ .

Beispiel

```
>K

All/Present(A/P)?A

Ready F0
>_
```

Löschen aller Dateien im gesamten Dateienbereich.

13-7-8. B (Leerstelle)

B

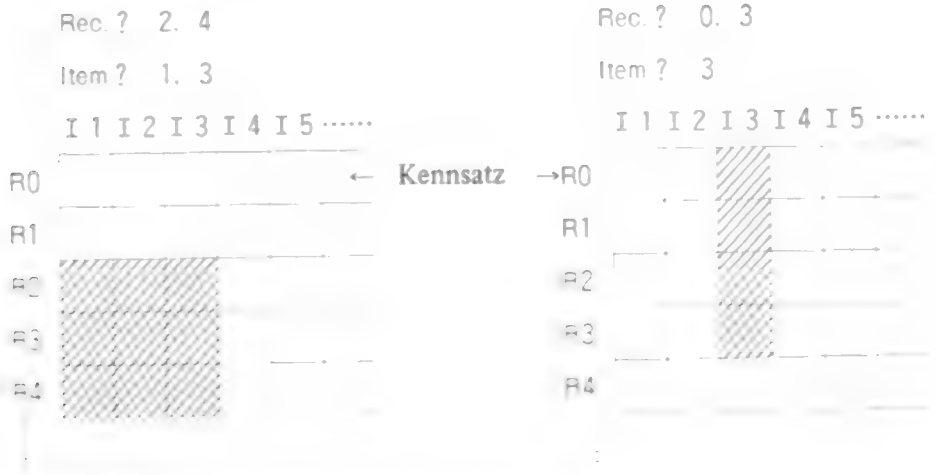
Funktion

Einschieben von Leerstellen in Datenzellen.

Erklärungen

(1) Durch diesen Befehl werden Leerstellen in die Datenzellen in den bezeichneten Datensätzen bzw. Posten eingeschoben, und der vorhandene Zelleninhalt wird gelöscht.

(2) Bezeichnen Sie nach Eingabe des Befehls B zuerst den Datensatzbereich und dann den Postenbereich.

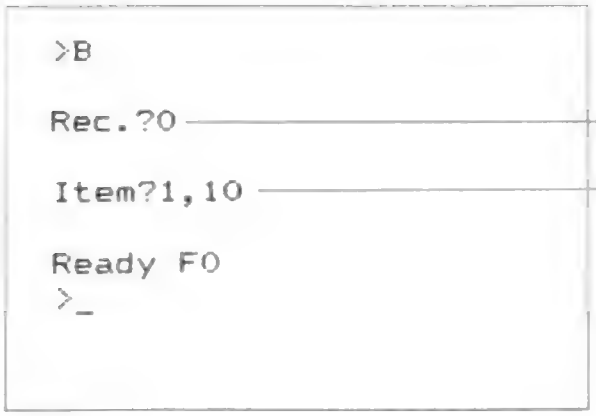


(3) Wenn der für Datensatze und Posten bezeichnete Bereich nicht im Bereich der gegenwartig existierenden Datensatze und Posten liegt, so wird ein Fehler verursacht.

(4) Der Kennsatz kann auch als Datensatz bezeichnet werden. Wenn der Kennsatz bezeichnet wird, so werden der Datenwortname, der Berechnungsausdruck und das Format auch gelöscht. (Beachten Sie jedoch, daß der Typ nicht gelöscht, sondern auf N zurückgestellt wird.)

(5) Die Postennamen (die Postennamen in den Kennsatzen) konnen auch als Posten bezeichnet werden. Geben Sie für diese Bezeichnung den Postennamen (in Anführungszeichen eingeschlossen) anstatt der postennummer ein. Wenn der Name nicht existiert, so wird ein Fehler verusacht.

Beispiel



Bezeichnung des Datensatzes 0 (Kennsatz).

Eingabe von Leerstellen für die Posten 1 bis 10.

13-7-9. S (Sortieren)

S

Funktion

Sortieren bezeichneter Zellen in einem Postens.

Erklärungen

(1) Durch diesen Befehl werden die Zellen eines bezeichneten Postens im Bereich der bezeichnet Datensatznummern in abfallender oder ansteigender Reihenfolge sortiert.

(2) Bezeichnen Sie nach Eingabe des Befehls S den Schlüsselpunkt (Postenname oder Postennummer), für den Sortieren durchzuführen ist. Bezeichnen Sie dann "D" (abfallend) für abfallende Reihenfolge oder "U" (ansteigend) für ansteigende Reihenfolge. Verwenden Sie die Datensatznummern zum Bezeichnen des Bereichs, für den Sortieren durchzuführen ist. Geben Sie in Bezug auf Speicherbewegung (Mem. Move (Y/N) ? "N" ein, wenn das Ergebnis des Sortierens angezeigt, aber nicht im Speicher bewegt werden soll, oder geben Sie "Y" ein, wenn das Ergebnis im Speicher bewegt, aber nicht angezeigt werden soll.

(3) Wenn "N" bezeichnet wird, so besteht der nächste Schritt darin, festzulegen ob das Ergebnis auf dem Drucker ausgedruckt werden sollte oder nicht.

(4) Schließen Sie einen Postennamen für Eingabe als Schlüsselpunkt in Anführungszeichen ein. Wenn der Postenname nicht existiert, so wird ein Fehler verursacht.

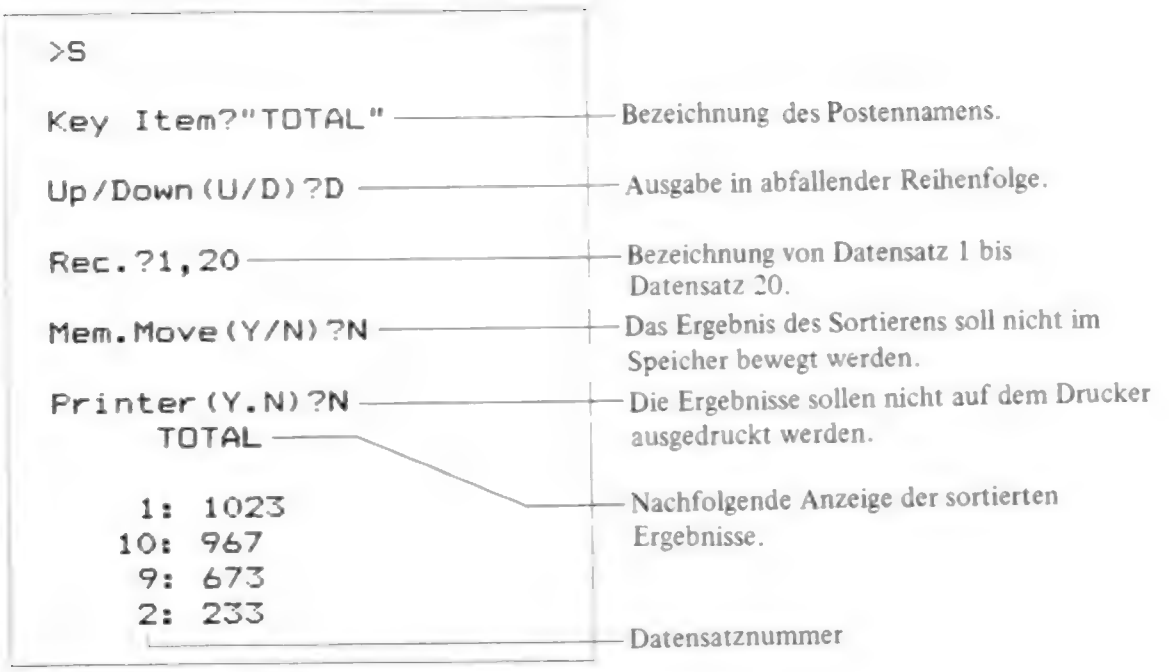
(5) Wenn der Datensatzbereich den gegenwärtig definierten Bereich überschreitet, so wird ein Fehler verursacht.

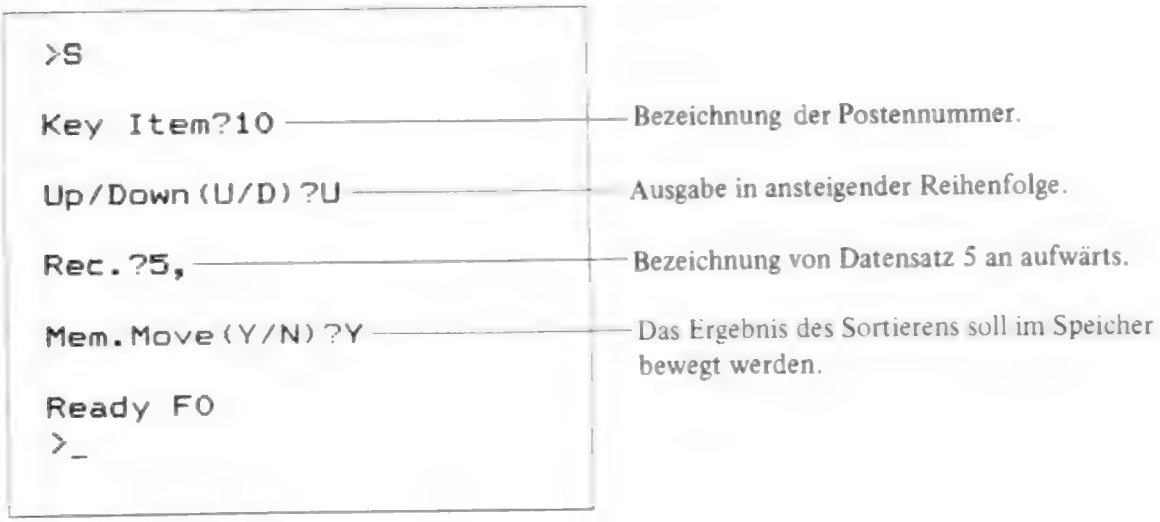
Rec.? 1,30 . . . . . zeigt die Datensatznummern von 1 bis 30 an.

Rec.? 5, ... zeigt die Datensatznummern von 5 bis zur höchsten gegenwärtigen Nummer an.

* Beachten Sie, daß Rec. 0 nicht bezeichnet werden kann.

Beispiele


>S
Key Item?"TOTAL"
Bezeichnung des Postennamens.
Up/Down(U/D)?D
Ausgabe in abfallender Reihenfolge.
Rec.?1,20
Bezeichnung von Datensatz 1 bis Datensatz 20.
Mem.Move(Y/N)?N
Das Ergebnis des Sortierens soll nicht im Speicher bewegt werden.
Printer(Y.N)?N
Die Ergebnisse sollen nicht auf dem Drucker ausgedruckt werden.
TOTAL
Nachfolgende Anzeige der sortierten Ergebnisse.
1: 1023
10: 967
9: 673
2: 233
Datensatznummer


>S
Key Item?10
Bezeichnung der Postennummer.
Up/Down(U/D)?U
Ausgabe in ansteigender Reihenfolge.
Rec.?5,
Bezeichnung von Datensatz 5 an aufwärts.
Mem.Move(Y/N)?Y
Das Ergebnis des Sortierens soll im Speicher bewegt werden.
Ready F0
>_

13-7-10. F (Finden)

```
F
```

Funktion

Durchführung von Auffinden nach Bedingung und Ausgabe der Datensatze, die der Bedingung entsprechen.

Erklärungen

(1) Durch diesen Befehl werden die Indexpunkte (Inhalt von Posten 1) der Datensätze angezeigt oder ausgedruckt, die der bezeichneten Bedingung entsprechen.

(2) Bezeichnen Sie nach Eingabe des Befehls F den Datensatzbereich, den Bedingungsausdruck und ob Druckerausgabe gewünscht ist.

(3) Der Datensatzbereich muß im Bereich von Datensatznummer 1 bis zur gegenwärtig definierten Datensatznummer liegen: Der Datensatz 0 kann nicht bezeichnet werden.

Rec. ?5, 15 . . . . . Auffinden wird von Datensatz 5 bis Datensatz 15 durchgeführt

Rec. ?1, ... Auffinden wird von Datensatz 1 ab durchgeführt.

(4) Der als Bedingung für Auffinden verwendete Bedingungsausdruck sollte als Vergleichsausdruck oder als logischer Ausdruck eingegeben werden.
Im Vergleichsausdruck können die IT-Funktion, die RC-Funktion und Ausdrücke (numerische Ausdrücke oder Zeichenausdrücke) verwendet werden. Vergleichsausdrücke und boolesche Ausdrücke können durch Verwendung der logischen Operatoren AND, OR, NOT und XOR kombiniert werden.

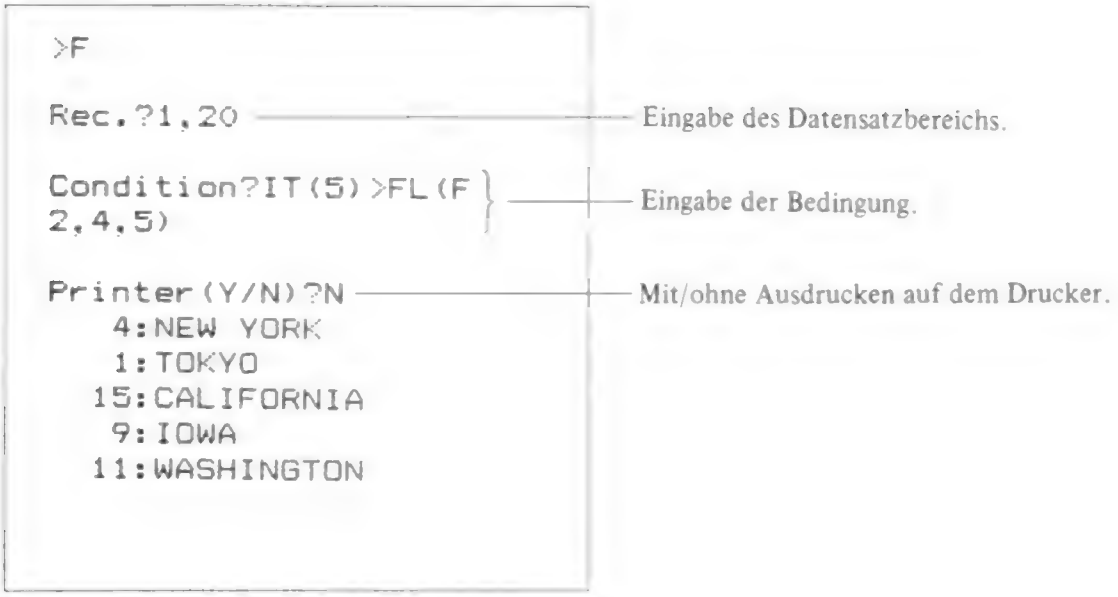
IT(10) > 100 . . . . . Auffinden aller Daten von Posten 10, die größer als 100 sind.

IT("TOTAL") = < 50 . . . . . Auffinden aller Daten des Postens mit dem Namen "TOTAL", die kleiner oder gleich 50 sind.

IT(2) > 60 AND IT(3) > 80 . . . . Auffinden aller Daten von Posten 2 über 60 und aller Daten von Posten 3 größer als 80.

(5) Geben Sie "Y" für "Printer (Y/N) ?" ein, wenn das Ergebnis auf dem Drucker ausgedruckt werden soll.

Beispiel


>F
Rec.?1,20
Condition?IT(5)>FL(F
2,4,5)
Printer(Y/N)?N
4:NEW YORK
1:TOKYO
15:CALIFORNIA
9:IOWA
11:WASHINGTON
Eingabe des Datensatzbereichs.
Eingabe der Bedingung.
Mit/ohne Ausdrucken auf dem Drucker.

13-7-11. J (Sprung)

```
J
```

Funktion

Sprung zur bezeichneten Datenzelle und Einstellen des Status für Datenmodifikation oder Eingabe.

Erklärungen

(1) Durch diesen Befehl erfolgt Sprung zur bezeichneten Datenzelle und Anzeige des Inhalts. Anschließend kann die Anzeige mit den Tasten , **SHIFT** + , **SHIFT** + , **SHIFT** + und **SHIFT** + bewegt werden.

(2) Während der Inhalt angezeigt wird, ist die Redigierbetriebsart eingeschaltet. Wenn deshalb die Taste nach alphanumerischen Tasten oder Symboltasten (ausgenommen die oben angeführten Tasten für Bewegung des Positionsanzeigers) gedrückt wird, so werden die angezeigten Zeichen in der Zelle aufgezeichnet.

(3) Bezeichnen Sie nach der Eingabe dieses Befehls die Position durch den Datensatz und die Postennummer.

(4) Wenn der bezeichnete Datensatz bzw. Posten nicht existiert, so wird ein Fehler verursacht.

Beispiel

```
>J
Rec.?6
Item?3
6-3 :123_
```

Bezeichnung des Datensatzes.

Bezeichnung des Postens.

13-7-12. L (Auflisten)

```
L
```

Funktion

Ausdrucken des Inhalts der gegenwärtigen Datei auf dem Drucker.

Erklärungen

(1) Durch diesen Befehl wird der Inhalt einer Datei auf dem Drucker ausgedruckt.

(2) Der ausgedruckte Inhalt besteht aus Dateiname, Anzahl der Datensätze, Anzahl der Posten, Inhalt des Kennsatzes (Postenname, Typ, Ausdruck, Format) und Daten in jeder Datenzelle.

(3) Wenn der Drucker nicht angeschlossen ist oder wenn der Netzadapter nicht verwendet wird, so wird ein Fehler verursacht.

Beispiel

```
>L

Ready F0
>_
```

```
File Name:SALES

  Number of Rec.:6
  Number of Item:4


Label Record
Item1:NAME
   Type(N/S)   :S
   Expression ?
   Format      :&           &

Item2:SALES
```

```
5-1 :WASHINGTON
5-2 :234
5-3 :92
5-4 :13.5652

6-1 :TOTAL
6-2 :SUMRC(1,5)
6-3 :SUMRC(1,5)/5
6-4 :SUMRC(1,5)
```

13-7-13. T (Tabelle)

T

Funktion

Anzeige und Ausdrucken der Dateidaten in Tabellenformat/nach Berechnung, falls ein Ausdruck bezeichnet worden ist.

Erklärungen

(1) Durch diesen Befehl werden alle Daten in den gegenwärtig definierten Dateien ausgegeben. Wenn ein Ausdruck in der Datei geschrieben ist, so wird das Ergebnis der Berechnung angezeigt bzw. ausgedruckt.

(2) Das Berechnungsergebnis wird ausgedruckt, aber es wird nicht im Speicher gespeichert. Führen Sie deshalb den Befehl C aus, wenn das Ergebnis im Speicher gespeichert werden soll.

(3) Bezeichnen Sie nach der Ausführung des Befehls T den Datensatzbereich bzw. den Postenbereich für die Ausgabe. Wenn der bezeichnete Datensatzbereich bzw. Postenbereich den gegenwartig definierten Bereich uberschreitet, so wird ein Fehler verursacht.

(4) Wenn die Ausgabe auf dem Drucker ausgedruckt werden soll, so geben Sie "Y" als Antwort auf "Printer (Y/N) ?" ein.

(5) Die Ausgabe wird im bezeichneten Format angezeigt bzw. ausgedruckt. Beachten Sie jedoch, das die Ausgabe schwierig zu lesen ist, wenn sie mit mehr als 20 Spalten formatiert ist oder die Breite des Druckers überschreitet.

Beispiel

```
>T

Rec.?0,
Item?1,
Printer(Y/N)?Y

Ready F0
>_
```

Eingabe des Datensatzbereichs (von Datensatz 1 bis zum Ende)

Eingabe des Postenbereichs (von Posten 1 bis zum Ende)

Mit Ausdrucken auf dem Drucker.

NAME	SALES	GROWTH RATE	RATIO
NEW YORK	345	123.00	20.00
TOKYO	567	163.00	32.87
CALIFORNIA	123	97.00	7.13
IOWA	456	104.00	26.43
WASHINGTON	234	92.00	13.57
TOTAL	1,275	115.80	100.00

13-7-14. C (Berechnen)

```
C
```

Funktion

Berechnung des Wertes eines in einer Datei enthaltenen Ausdrucks und Zuweisen des Ergebnisses zur entsprechenden Datenzelle.

Erklärungen

(1) Durch diesen Befehl wird der Wert des Ausdruckes in einem Kennsatz oder einer Datenzelle berechnet, und das Ergebnis wird der entsprechenden Datenzelle zugewiesen.

(2) Bezeichnen Sie nach der Eingabe des Befehls C den Datensatzbereich und den Datenwortbereich.

(3) Wenn der bezeichnete Datensatzbereich bzw. Datenwortbereich den gegenwärtig definierten Bereich überschreitet, so wird ein Fehler verursacht.

(4) Wenn ein Ausdruck in einer Datenzelle geschrieben ist, so wird das Rechnungsergebnis in die Zelle geschrieben. Der Ausdruck wird deshalb automatisch überschieben und gelöscht.

(5) Wenn Daten geändert werden und Neuberechnung nach Durchführung des Befehls C erforderlich ist, so verfahren Sie wie folgt.

(a) Wenn der Ausdruck in einem Kennsatz bezeichnet ist:

1) Führen Sie den Befehl B für die von dem Ausdruck betroffenen Posten durch.
(Da die Berechnung für den Ausdruck nicht durchgeführt wird, wenn die Daten in den Objektpunkten nicht Leerstellen sind.)

2) Führen Sie den Befehl C erneut durch.

(b) Wenn der Ausdruck in einer Zelle bezeichnet ist:

1) Springen Sie zu dieser Zelle und schreiben Sie den Ausdruck erneut.

2) Springen Sie wieder zum Befehl C zurück.

Beispiel

```
>C
Rec.?2,5 ——— Eingabe des Datensatzbereichs.
Item?1,  ——— Eingabe des Postenbereichs.
Ready F0
>_
```

13-7-15. P (Schreiben)

```
P
```

Funktion

Ausgabe von Daten zu einem externen Gerät.

Erklärungen

(1) Durch diesen Befehl werden Daten in den gegenwärtig definierten Dateien zu einem externen Gerät ausgegeben.

(2) Als externes Gerät kann FDD, RS-232C oder CMT bezeichnet werden.
F: FDD
R: RS-232C
C: CMT

(3) Wenn FDD oder CMT bezeichnet wird, so werden die gegenwärtig definierten Dateinamen in dem entsprechenden Gerät gespeichert.

Beispiel

```
>P

Out(F/S/C)?C

Ready F0
>_
```

Bezeichnung des Tonbandgerätes.

13-7-16. G (Lesen)

```
G
```

Funktion

Lesen von Daten von einem externen Gerät.

Erklärungen

(1) Durch diesen Befehl werden Daten von einem externen Gerät gelesen, und es wird eine Datei erstellt.

(2) Wenn eine Datei im gegenwärtigen Dateibereich vorhanden ist, so wird ein Fehler verursacht.

(3) Als externes Gerät kann FDD, RS-232C oder CMT bezeichnet werden.
F: FDD
S: RS-232C
C: CMT

(4) Wenn FDD oder CMT bezeichnet wird, so wird der Dateiname angefordert. Geben Sie den Namen der zu lesenden Datei ein (maximal acht Zeichen).

(5) Wenn das externe Gerät ein Kassettentonbandgerät ist und der Dateiname ausgelassen wird, so wird die zuerst gelesene Datei eingelesen.

(6) Wenn das externe Gerät FDD ist und der Dateiname ausgelassen wird, so wird die Datei gelesen, deren Name aus lauter Leerstellen besteht (die Datei ohne Namen). Wenn keine solche Datei vorhanden ist, so wird ein Fehler verursacht.

(7) Datenaustausch zwischen zwei FP-200 über RS-232C unter Verwendung der Befehle P und G ist nicht möglich.

Beispiel

```
>G
In(F/S/C)?C
File Name?TEST
TEST            F

Ready F3
>_
```

Bezeichnung des Tonbandgerätes.

Eingabe des Dateinamens.

13-8. CETL -Verwaltungsfunktionen

Es gibt zwei Methoden, um Daten in Dateien einzuschieben, die in CETL-Betriebsart erstellt worden sind. Die Daten können entweder manuell direkt in die Datenzelle eingegeben werden, oder die Daten können unter Verwendung von Ausdrucken oder BASIC-Programmen berechnet und automatisch gesetzt werden. Die Dateiverwaltungsfunktionen sind nutzlich für Eingabe von Daten mit der letzteren Methode.

Beachten Sie bei Verwendung der auf den folgenden Seiten erklärten Verwaltungsfunktionen die folgenden Posten.

(1) Wenn ein Argument für eine Funktion mit einem nicht tatsächlich existierenden Datensatz oder Posten bezeichnet wird, so wird ein Fehler verursacht.

(2) Numerische Ausdrücke können als Argument einer Funktion (Datei, Datensatz oder Posten) verwendet werden. In diesem Fall wird das Ergebnis der Bewertung des numerischen Ausdrucks auf einen ganzzahligen Wert gerundet.

(3) In eckige Klammern eingeschlossene Elemente sind optional.

(4) Die fett gedruckten Buchstaben müssen wie gezeigt eingegeben werden.

13-8-1. RC-Funktion

Funktion

Rückgabe der gegenwärtigen Datensatznummer.

Erklärungen

(1) Diese Funktion gibt die gegenwärtige Datensatznummer.

(2) Nach Ausführung eines CETL-Befehls wird die Nummer auf 0 eingestellt.

(3) Für einen Kennsatz wird RC=0 zurückgegeben.

13-8-2. IT-Funktion

IT

Funktion

Rückgabe der gegenwärtigen Postennummer.

Erklärungen

(1) Diese Funktion gibt die gegenwärtige Postennummer.

(2) Nach Ausführung eines CETL-Befehls wird die Nummer auf 1 eingestellt.

13-8-3. RC( )-Funktion

RC (Datensatzbezeichnung)

Datensatznummer oder Datensatzname

Funktion

Ruckgabe des Inhalts der Datenzelle im gegenwartigen Posten, der dem bezeichneten Datensatz entspricht.

Parameter

Datensatznummer: $0 \leqq$ Datensatznummer $\leqq$ Anzahl der definierten Datensätze
Bei Verwendung eines numerischen Ausdrucks:
$0 \leqq$ Datensatzdefinition $\leqq$ Anzahl der definierten Datensätze, bewertet als ganze Zahl.

Datensatzname: Ein Datensatzname in einem Indexpunkt
(Indexpunkt = Datenwort Nr. 1)

Erklärungen

(1) Diese Funktion gibt den Dateninhalt in der Datenzelle im gegenwärtigen Posten in der gegenwartig definierten Datei, die durch den bezeichneten Datensatz bezeichnet ist, zurück.

(2) Eine Datensatznummer kann durch einen numerischen Ausdruck bezeichnet werden.

(3) Ein Datensatzname kann durch Zeichenkonstanten und durch Zeichenvariablen bezeichnet werden.

Beispiel

```
RC(3)
RC ("TOKYO")
```

13-8-4. IT( )-Funktion

IT (Postenbezeichnung)
Datenwortnummer oder Datenwortname

Funktion

Rückgabe des Inhalts der Datenzelle im gegenwartigen Datensatz entsprechend dem bezeichneten Posten.

Parameter

Postennummer: $1 \leqq$ Postennummer $\leqq$ Anzahl der definierten Posten
Bei Verwendung eines numerischen Ausdrucks:
$1 \leqq$ Postendefinition $\leqq$ Anzahl der definierten Posten
bewertet als ganze Zahl.
Postenname: Ein Postenname in einem Kennsatz.

Erklärungen

(1) Diese Funktion gibt den Inhalt der Datenzelle im gegenwartigen Datensatz innerhalb der gegenwartig definierten Datei, angezeigt durch den bezeichneten Posten, wieder.

(2) Eine Postennummer Kann unter Verwendung eines numerischen Ausdrucks bezeichnet werden.

(3) Ein Teilname kann durch Verwendung von Zeichenkonstanten und Zeichenvariablen bezeichnet werden.

Beispiel

```
IT (5)
IT ("sales")
IT (A$ + B$)
```

13-8-5. FL-Funktion

FL ([Dateispezifikation] , [Datensatzspezifikation] , [Postenspezifikation])

- [Dateispezifikation]: Dateinummer oder Dateiname
- [Datensatzspezifikation]: Datensatznummer oder Datensatzname
- [Postenspezifikation]: Postennummer oder Postenname

Funktion

Lesen des Inhalts der durch den gegebenen Datensatz und Posten bezeichneten Datenzelle oder Zuordnung von Daten zu der durch den gegebenen Datensatz und Posten bezeichneten Datenzelle in der bezeichneten Datei.

Parameter

Dateispezifikation

Dateinummer: $0 \leqq$ Dateinummer < 10 (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet).

Dateiname: Ein Dateiname.

Datensatzspezifikation

Datensatznummer: $0 \leqq$ Datensatznummer $\leqq$ Anzahl der definierten Datensätze (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet).

Datensatzname: Ein Datensatzname in einem Indexpunkt.

Postenspezifikation

Postennummer: $0 \leqq$ Postennummer $\leqq$ Anzahl der definierten Datenwörter (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet).

Postenname: Ein Postenname in einem Kennsatz.

Erklärungen

(1) Diese Funktion liest Daten in der Datenzelle, die der bezeichneten Datei und dem bezeichneten Datensatz und Posten entspricht, oder sie schiebt Daten in diese Datenzelle ein.

(2) Wenn in BASIC-Betriebsart eine Zuordnungsanweisung mit der FL-Funktion auf der linken Seite durchgeführt wird, so können Daten zu der Datenzelle von bezeichneter Datei, Datensatz und Posten zugeordnet werden. (In der CETL-Betriebsart kann Zuordnung nicht erfolgen.)

(3) Die Definitionen für Datei, Datensatz und Posten können ausgelassen werden. Wenn einer dieser Punkte ausgelassen wird, so wird automatisch die gegenwärtige Position (die Datenzelle, die bei Eintritt in den Redigierbetrieb durch Druck auf die Taste **PF0** angezeigt wird) bezeichnet.

(4) Ein Dateiname, ein Datensatzname und ein Datenpunktname können durch Zeichenkonstanten und Zeichenvariablen bezeichnet werden.

(5) Wenn ein Kennsatz bezeichnet wird, so wird entsprechend dem Datentyp des festgelegten Punktes 0 oder eine Leerstelle gelesen.

Beispiel

```
PRINT FL (0, 1, 2)
Z = FL (0, N + 1, M + J)
[nur BASIC Betriebsart]
FL (0, 5, 7) = 1234
FL (1, 0, 3) = "DATA 3"
```

13-8-6. SUMRC-Funktion

SUMRC (Datensatzspezifikation 1 , Datensatzspezifikation 2)

Datensatzspezifikation 1: Datensatznummer oder Datensatzname

Datensatzspezifikation 2: Datensatznummer oder Datensatzname

Funktion

Berechnung der Summe der Daten im gegenwärtigen Datenwort in Zellen innerhalb des bezeichneten Datensatzbereichs.

Parameter

Datensatzspezifikation 1

Datensatznummer: $0 \leqq$ Datensatzdefinition 1 $\leqq$ Anzahl der definierten Datensätze (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet)

Datensatzname: Ein Datensatzname in einem Indexpunkt.

Datensatzspezifikation 2

Datensatznummer: $0 \leqq$ Datensatzdefinition 2 $\leqq$ Anzahl der definierten Datensätze (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet)

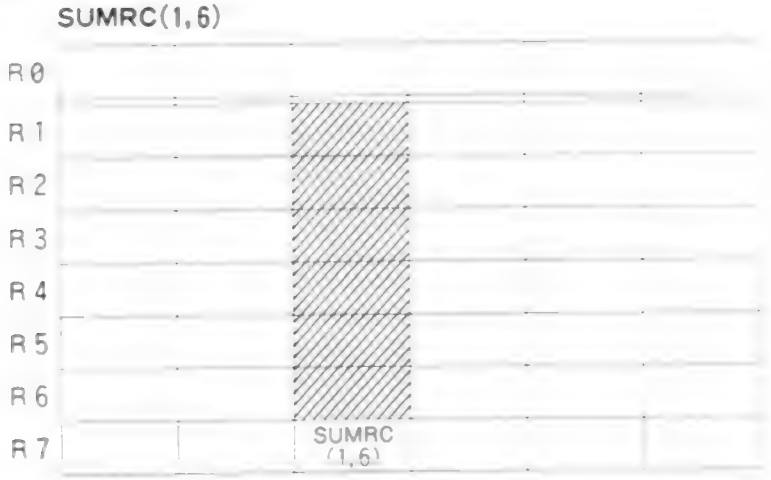
Datensatzname: Ein Datensatzname in einem Indexpunkt.

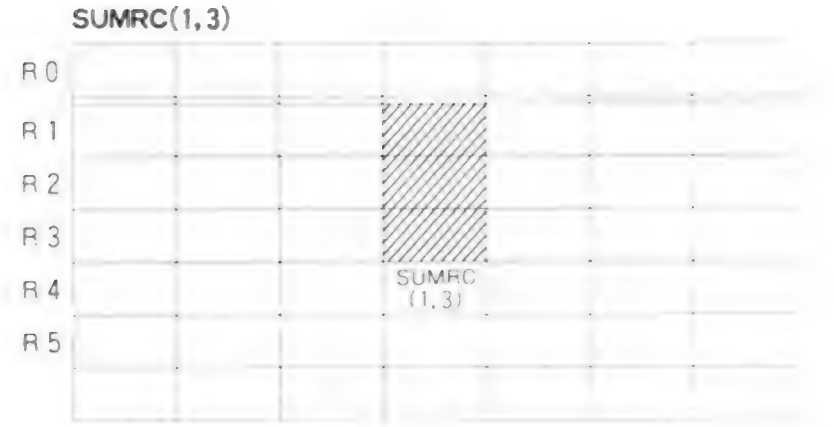
Die Datensatzspezifikation 1 muß kleiner als oder gleich Datensatzspezifikation 2 sein.

Erklärungen

Diese Funktion berechnet die Summe der Daten in dem gegenwärtigen Posten innerhalb des durch die beiden Datensatzspezifikationen gegebenen Bereichs.

Beispiel




SUMRC(1,3)
R0
R1
R2
R3
R4
R5
SUMRC
(1,3)

13-8-7. SUMIT-Funktion

SUMIT (Postenspezifikation 1 , Postenspezifikation 2)

Postenspezifikation 1: Postennummer oder Postenname
Postenspezifikation 2: Postennummer oder Postenname

Funktion

Berechnung der Summe der Daten im gegenwärtigen Datensatz in Zellen innerhalb des durch die beiden Postenspezifikationen gegebenen Bereichs.

Parameter

Postenspezifikation 1

Posten: $1 \leqq$ Postennummer 1 $\leqq$ Anzahl der bezeichneten Posten (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet)

Postenname: Ein Postenname in einem Kennsatz

Postenspezifikation 2

Posten: $1 \leqq$ Postennummer 2 $\leqq$ Anzahl der bezeichneten Posten (bei Verwendung eines numerischen Ausdrucks als ganze Zahl bewertet)

Postenname: Ein Postenname in einem Kennsatz

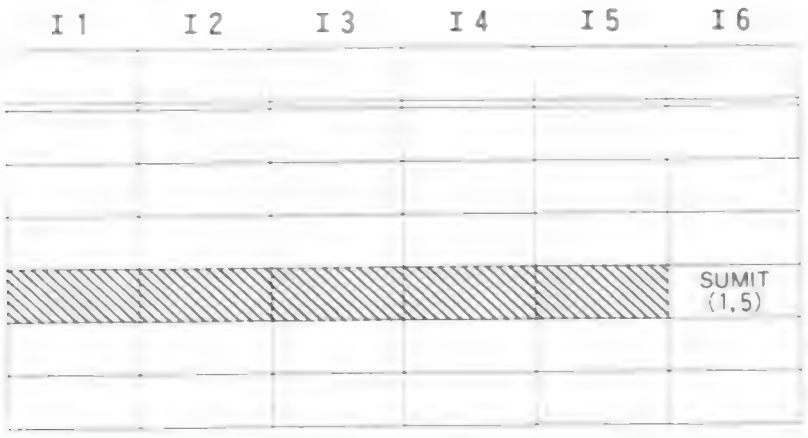
Die Postenspezifikation 1 muß kleiner als oder gleich Postenspezifikation 2 sein.

Erklärung

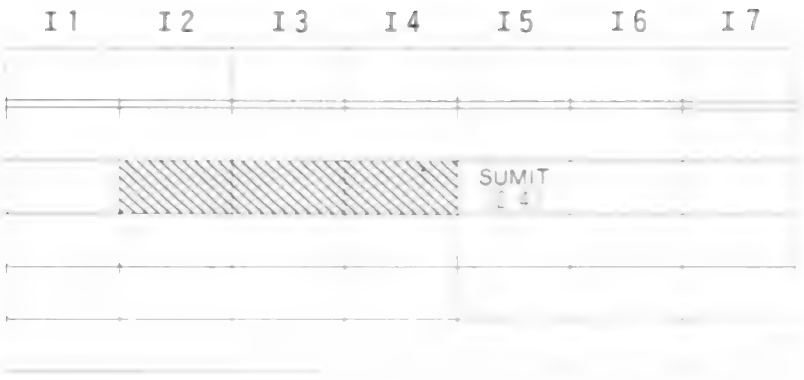
Diese Funktion berechnet die Summe der Daten im gegenwärtigen Datensatz in Zellen innerhalb des durch die beiden Postenspezifikationen gegebenen Bereichs.

Beispiel

SUMIT(1,5)



SUMIT(2,4)



13-9. CETL und BASIC

Wenn in der CETL-Betriebsart ein Ausdruck zu kompliziert ist, um ihn in eine Datenzelle zu schreiben, oder wenn eine Bedingungsberechnung erwünscht ist, so konnen BASIC-Programme bequem von der CETL-Datenzelle abgerufen werden, um solche Aufgaben auszuführen.
Diese Verbindung zwischen CETL und BASIC wird als Verknüpfung bezeichnet.

PROG **Programmbereichsnummer ; Ausdruck**

Eine Stelle von 0 bis 9

Dieser PROG-Befehl sollte in eine Datenzelle oder in einen Ausdruck im Kennsatz geschrieben werden. Das BASIC-Programm in dem durch den PROG-Befehl bezeichneten Programmbereich wird ausgeführt, wenn CETL-Befehle wie C, T, F und S ausgeführt wird, und anschließend kann das Ergebnis der BASIC-Programmausführung durch die im Ausdruck bezeichneten Variablen in die Zelle eingesetzt werden.

Beispiel

```
F0 1-4:PROG 0;A
```

```
P0 10 INPUT B
   20 A=B*50
   30 END
```

Wenn einer der Befehle C, T, F oder S ausgeführt wird, so wird das Programm im Programmbereich 0 (P0) ausgeführt, und der Wert für A (A = B x 50) wird zur Zelle (1-4) im Dateibereich 0 (F0) übertragen.

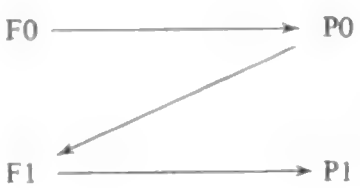
Hinweis: Verknüpfung zwischen CETL und BASIC ist zulässig bis zu doppelter Tiefe in beiden Richtungen (CETL zu BASIC, BASIC zu CETL). Wenn eine tiefere Verknüpfung versucht wird, so wird ein Fehler verursacht.

```
F0 1-1:PROG 0;A
```

```
P0 10 A=FL(1,1,1)
   20 END
```

```
F1 1-1:PROG 1;A
```

```
P1 10 A=123
```



13-10. Gemeinsame Verwendung sequentieller Dateien

Durch gemeinsame Verwendung sequentieller Dateien können Dateien zwischen CETL und BASIC ausgetauscht werden.
Auf diese Weise kann eine Datei in CETL erstellt werden, durch Ausführung des P-Befehls auf FDD oder CMT gespeichert werden, und durch Durchführung von sequentieller Eingabe von BASIC gelesen werden. Andererseits kann eine durch BASIC erstellte sequentielle Datei durch Ausführung des G-Befehls von CETL gelesen werden. Das folgende Programm illustriert dieses Verfahren.

Beispiel Erstellung einer CETL-Datei durch BASIC.

```
 10 CLEAR
 20 INPUT "File name=";A$
 30 INPUT "No. of Rec.=";B$
 40 INPUT "No. of Item=";C$
 50 OPEN A$ FOR OUTPUT AS #1
 60 PRINT #1,A$:PRINT #1,B$:PRINT #1,C$
 70 FOR R=1 TO VAL(C$)
 80 INPUT "Item name"+STR$(R)+"=";D$
 90 INPUT "Type=";E$
100 INPUT "Expression=";F$
110 INPUT "Format=";G$
120 PRINT #1,D$:PRINT #1,E$:PRINT #1,F$:PRINT #1,G$
130 NEXT R
140 FOR S=1 TO VAL(B$)
150 FOR T=1 TO VAL(C$)
160 INPUT " "+STR$(S)+"-"+STR$(T)+"=";H$
170 PRINT #1,H$
180 NEXT T
190 NEXT S
200 CLOSE:END
```

Hinweis: Beachten Sie bei Erstellung einer CETL-Datei durch Verwendung dieses Programms, daß alle ein Komma enthaltenden Ausdrücke (Ausdrücke, Datenzellen usw.) und Funktionen (SUMIT, SUMRC, FL) in Anführungszeichen eingeschlossen werden müssen und daß die Taste ↵ gedrückt werden muß.
FL (1, 2, 3) ⇒ "FL (1, 2, 3)" ↵

TASTENFELDEINGABETABELLE

Tastenoberseite	Normal		CAPS		GRAPH	CTRL
	Normal	SHIFT	Normal	SHIFT		
< ,	,	<	,	<	[redacted]	
= —	—	=	—	=	[redacted]	
> .	.	>	.	>	[redacted]	
? /	/	?	/	?	[redacted]	
0	0	0	0	0	秒	
! 1	1	!	1	!	市	
" 2	2	"	2	"	区	
# 3	3	#	3	#	町	
$ 4	4	$	4	$	村	
% 5	5	%	5	%	年	
& 6	6	&	6	&	月	
' 7	7	'	7	'	日	
(8	8	(	8	(	時	

Tastenoberseite	Normal		CAPS		GRAPH	CTRL
	Normal	SHIFT	Normal	SHIFT		
) 9	9	)	9	)	分	
	39	29	39	29	F6	
* :	:	*	:	*	▬	
	3A	2A	3A	2A	94	
+ ;	;	+	;	+	▮	
	3B	2B	3B	2B	89	
` @	@	`	@	`	▮	
	40	60	60	40	8A	00
A	A	a	a	A	◟	COMMAND MENU
	41	61	61	41	9E	01
B	B	b	b	B	▬	
	42	62	62	42	84	02
C	C	c	c	C	▬	
	43	63	63	43	82	03
D	D	d	d	D	◥	
	44	64	64	44	E6	04
E	E	e	e	E	◢	
	45	65	65	45	E4	05
F	F	f	f	F	◥	
	46	66	66	46	E7	06
G	G	g	g	G	●	
	47	67	67	47	EC	07
H	H	h	h	H	○	
	48	68	68	48	ED	08
I	I	i	i	I	♠	
	49	69	69	49	E8	09

Tastenoberseite	Normal		CAPS		GRAPH	CTRL
	Normal	SHIFT	Normal	SHIFT		
J	J	j	j	J	◆	
	4A	6A	6A	4A	EA	0A
K	K	k	k	K	♣	(HOME)
	4B	6B	6B	4B	EB	0B
L	L	l	l	L	▌	(CLS)
	4C	6C	6C	4C	8E	0C
M	M	m	m	M	▪	(RETURN)
	4D	6D	6D	4D	86	0D
N	N	n	n	N	▬	(NUMERIC)
	4E	6E	6E	4E	85	0E
O	O	o	o	O	♥	
	4F	6F	6F	4F	E9	0F
P	P	p	p	P	▮	
	50	70	70	50	8D	10
Q	Q	q	q	Q	◜	(DEL)
	51	71	71	51	9C	11
R	R	r	r	R	◣	(INS)
	52	72	72	52	E5	12
S	S	s	s	S	◞	(EDIT)
	53	73	73	53	9F	13
T	T	t	t	T	／	(▷)
	54	74	74	54	EE	14
U	U	u	u	U	✕	(◁)
	55	75	75	55	F0	15
V	V	v	v	V	▁	(△)
	56	76	76	56	83	16

Tastenoberseite	Normal		CAPS		GRAPH	CTRL
	Normal	SHIFT	Normal	SHIFT		
W	W	w	w	W		▽
	57	77	77	57	9D	17
X	X	x	x	X	■	
	58	78	78	58	81	18
Y	Y	y	y	Y	\	
	59	79	79	59	EF	19
Z	Z	z	z	Z	▬	
	5A	7A	7A	5A	80	1A
{ [	[	{	[	{	人	
	5B	7B	5B	7B	FD	1B
\	\		\		円	(→)
	5C	7C	5C	7C	F1	1C
}]	]	}	]	}	〒	(←)
	5D	7D	5D	7D	F8	1D
~ ^	^	~	^	~	▮	(↑)
	5E	7E	5E	7E	8B	1E
_		—		—		(↓)
		5F		5F	FE	
PF0	PF0	PF5	PF0	PF5	⌐	
					98	
PF1	PF1	PF6	PF1	PF6	⌐	
					99	
PF2	PF2	PF7	PF2	PF7	⌐	
					9A	
PF3	PF3	PF8	PF3	PF8	⌐	
					9B	

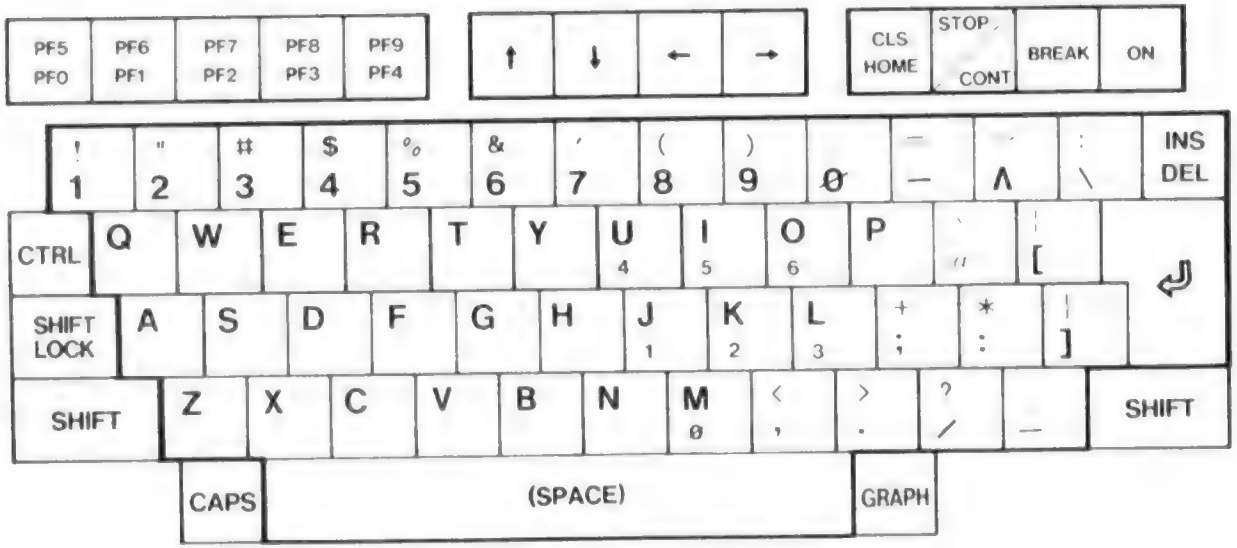
Tastenoberseite	Normal		CAPS		GRAPH	CTRL
	Normal	SHIFT	Normal	SHIFT		
PF4	PF4	PF9	PF4	PF9		
					96	
↑	↑	△	↑	△		
	1E	16	1E	16	95	
↓	↓	▽	↓	▽		
	1F	17	1F	17	91	
←	←	◁	←	◁		
	1D	15	1D	15	90	
→	→	▷	→	▷		
	1C	14	1C	14	93	
CLS HOME	HOME	CLS	HOME	CLS		
	0B	0C	0B	0C	92	
INS DEL	DEL	INS	DEL	INS		
	11	12	11	12	8F	
RETURN	RETURN	RETURN	RETURN	RETURN		
	0D	0D	0D	0D		

ERKLÄRUNG DER TASTENFUNKTIONEN

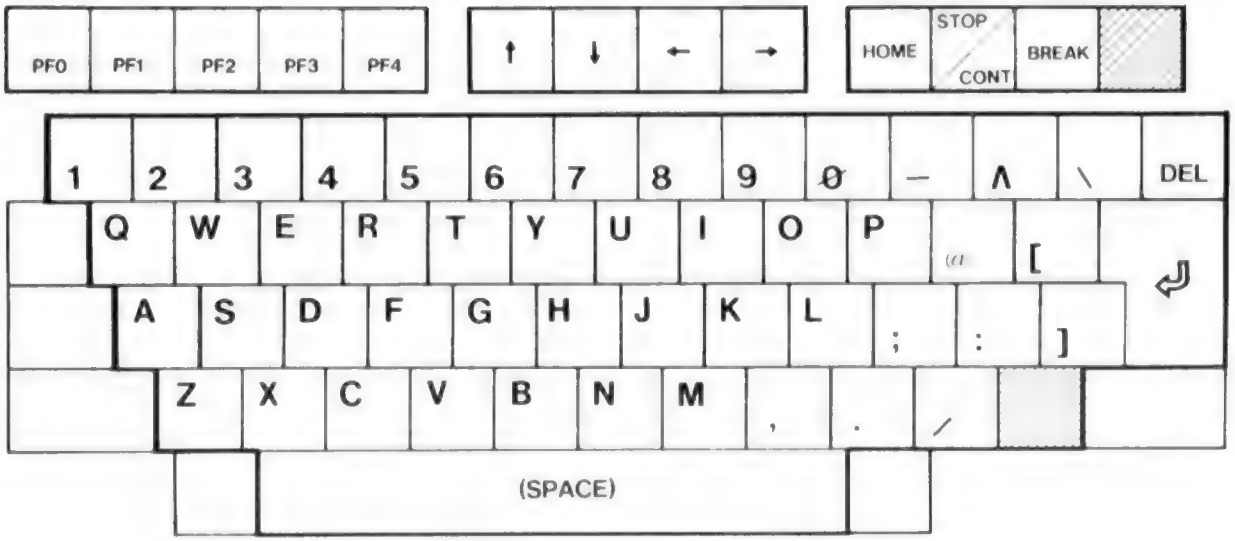
Funktion	Tastenbetätigung	Erklärung
INS	SHIFT + INS CTRL + R	Einschieben. Die auf den Positionsanzeiger folgenden Zeichen werden nach rechts bewegt, um Platz für einzuschiebende Zeichen zu schaffuen.
DEL	DEL CTRL + Q	Löschen. Das Zeichen an der Position des Positionsanzeigers wird gelöscht, und die folgenden Zeichen werden nach links bewegt.
HOME	CLS HOME CTRL + K	Bewegen des Positionsanzeigers zur linken oberen Ecke des Bildschirms.
CLS	SHIFT + CLS HOME CTRL + L	Löschen des Bildschirms und Bewegen des Positionsanzeigers zur linken oberen Ecke des Bildschirms.
→	→ CTRL + \	Bewegen des Positionsanzeigers nach rechts. Wenn sich der Positionsanzeiger am rechten Rand befindet, so wird er zum linken Rand der nächsten Zeile bewegt. Wenn sich der Positionsanzeiger an der rechten unteren Ecke befindet, so wird er zur linken oberen Ecke bewegt.
←	← CTRL +]	Bewegen des Positionsanzeigers nach links. Wenn sich der Positionsanzeiger am linken Rand befindet, so wird er zum rechten Rand der direkt vorhergehenden Zeile bewegt. Wenn sich der Positionsanzeiger an der linken oberen Ecke befindet, so wird er zur rechten unteren Ecke bewegt.
↑	↑ CTRL + ^	Bewegung des Positionsanzeigers nach oben. Wenn sich der Positionsanzeiger auf der obersten Zeile befindet, so wird er zur untersten Zeile bewegt.
↓	↓ CTRL + _	Bewegung des Positionsanzeigers nach unten. Wenn sich der Positionsanzeiger auf der untersten Zeile befindet, so wird er zur obersten Zeile bewegt.
COMMAND MENU	CTRL + A	Dies bedeutet das CETL-Befehlsmenü.
EDIT	CTRL + S	Auf Redigerbetrieb umschalten.
NUM	CTRL + N	Bestimmte Buchstabentasten können als Hilfszifferntasten verwendet werden.
▷	SHIFT + → CTRL + T	Bewegung zur Zelle unterhalb der gegenwärtigen Zelle in der CETL-Betriebsart.

Funktion	Tastenbetätigung	Erklärung
—	SHIFT + ← CTRL + U	Bewegung um eine Zelle nach links von der gegenwärtigen Zelle in der CETL-Betriebsart.
—	SHIFT + ↑ CTRL + V	Bewegung um eine Zelle nach links von der gegenwärtigen Zelle in der CETL-Betriebsart.
—	SHIFT + ↓ CTRL + W	Bewegung um eine Zelle nach oben von der gegenwärtigen Zelle der CETL-Betriebsart.
RETURN	RETURN CTRL + M	Diese Taste wird am Ende der Dateneingabe gedrückt. Der Inhalt einer logischen Zeile wird zum Rechner eingegeben.

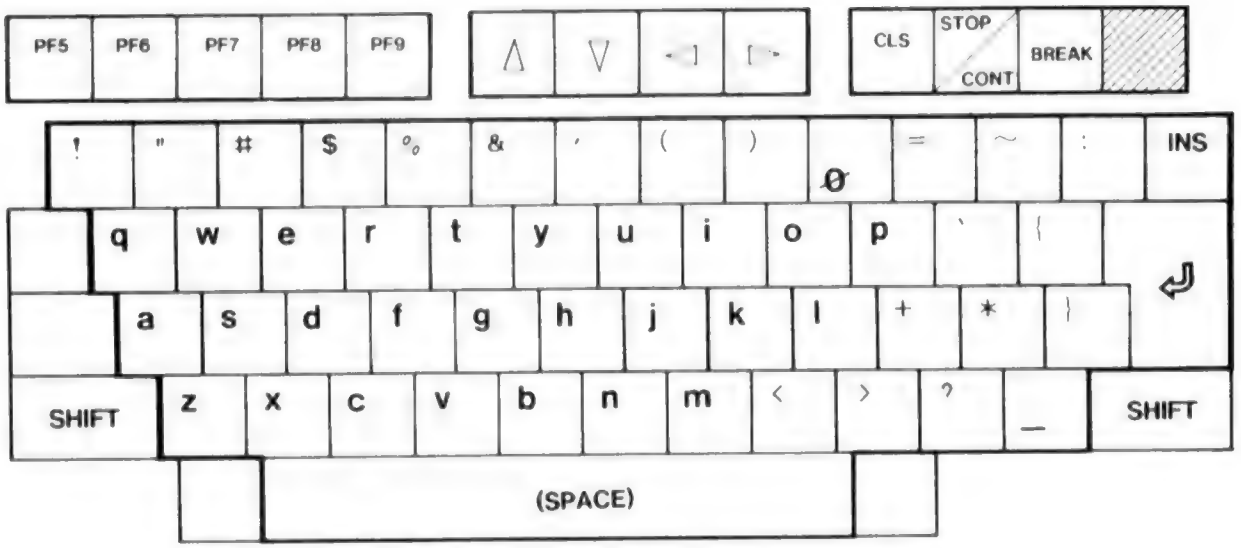
Tastatur-Layout



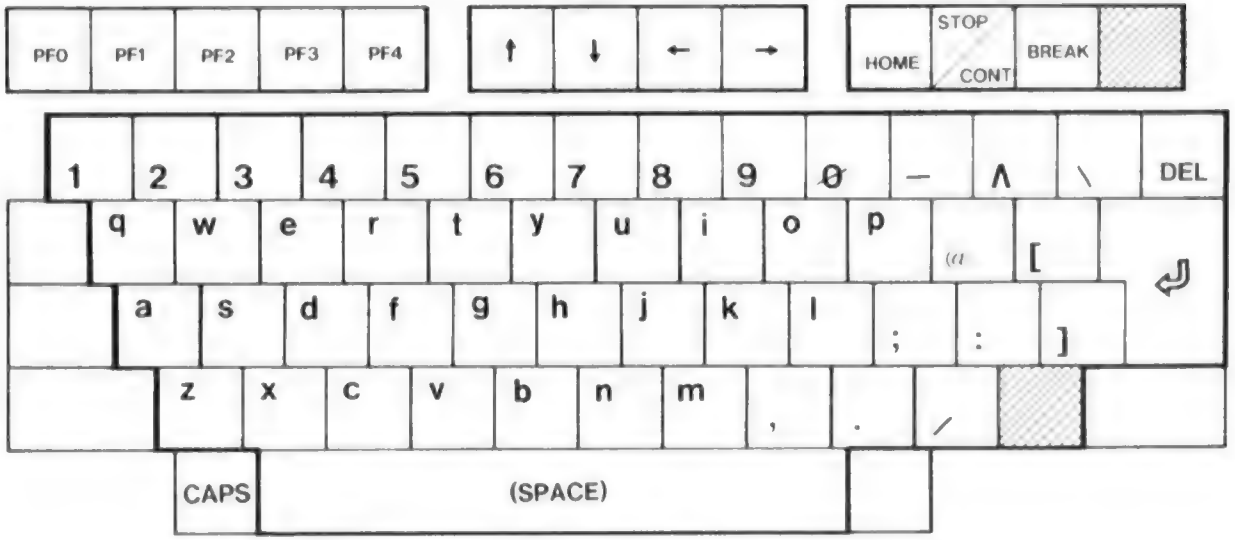
Normale Tastenfunktionen



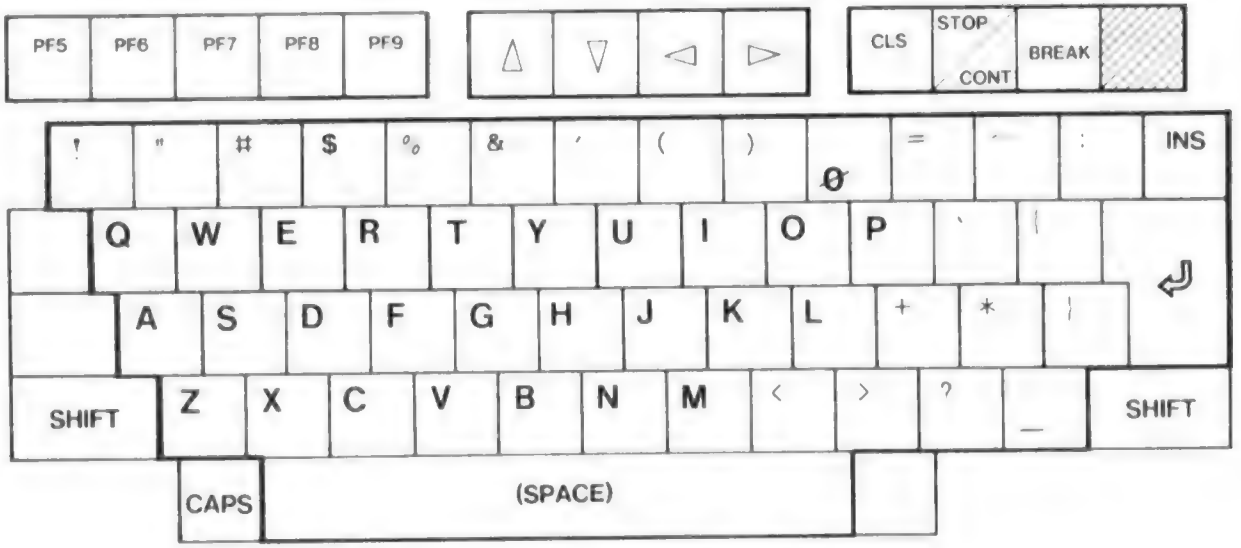
Umschalt-Tastenfunktionen



Tastenfunktionen bei Großbuchstaben-Umschaltsperre

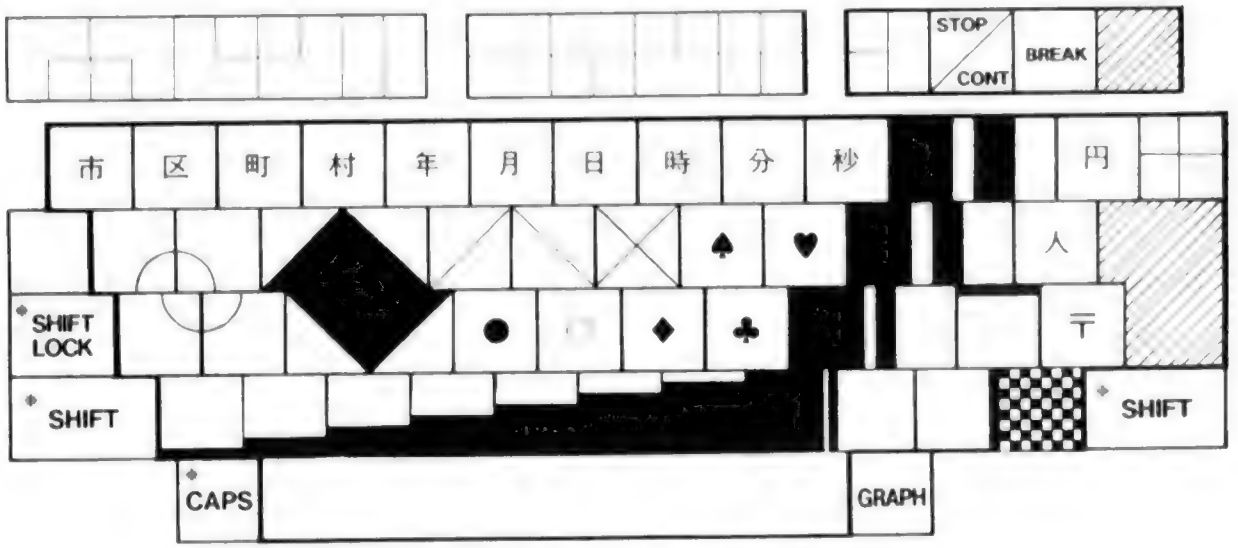


Tastenfunktionen in der Betriebsart Großbuchstaben-Umschaltsperre

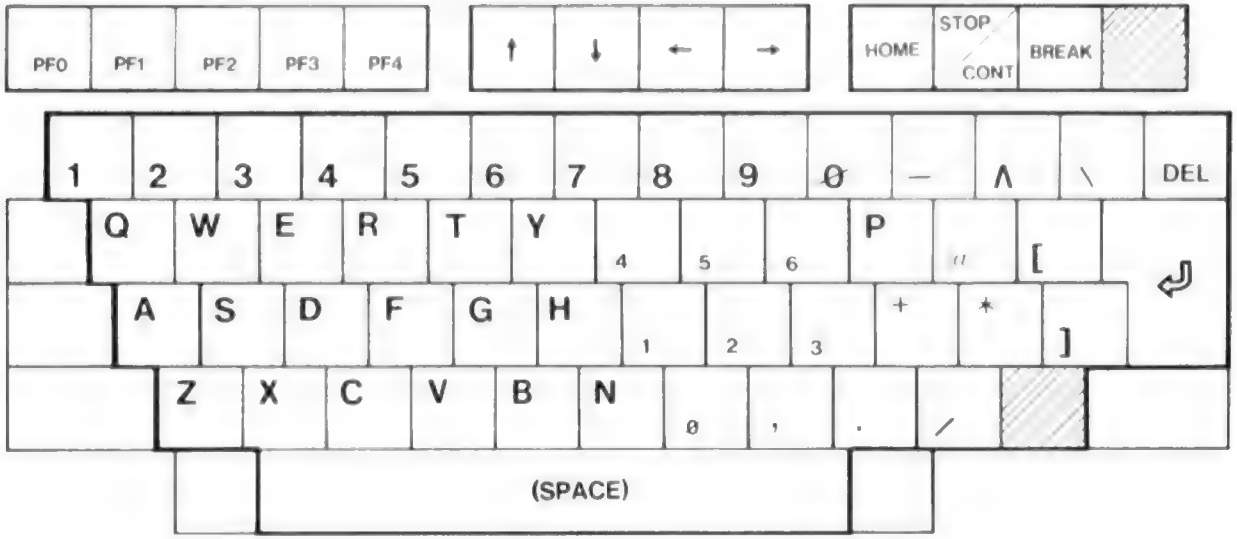


Tastenfunktionen in der Grafik-Betriebsart

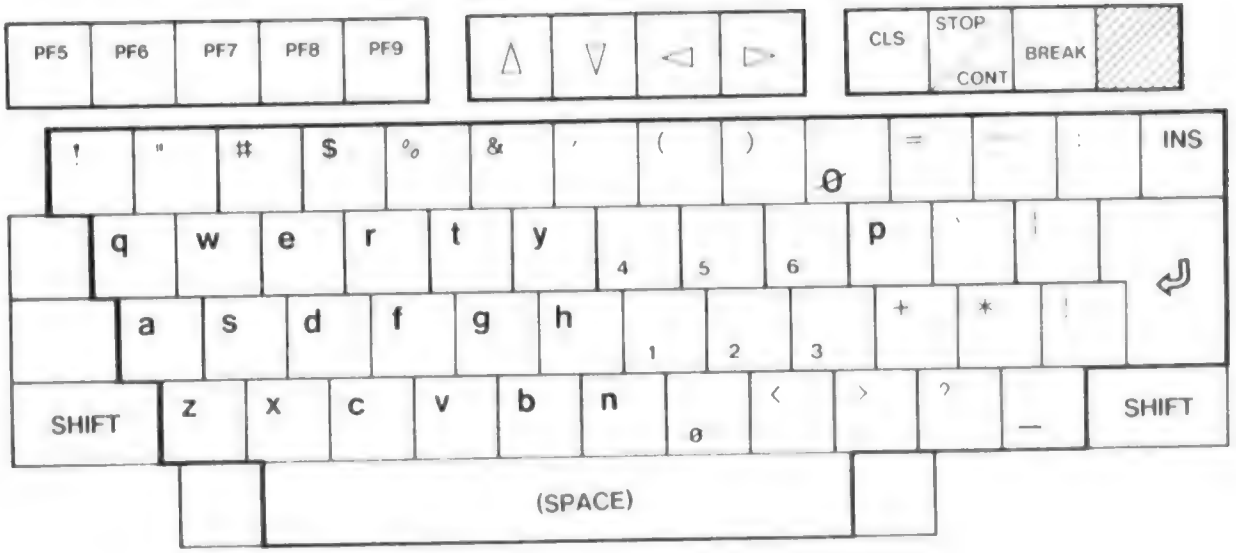
* Das gleiche gilt für den durch Kombination von CAPS und SHIFT erhaltenen Status.



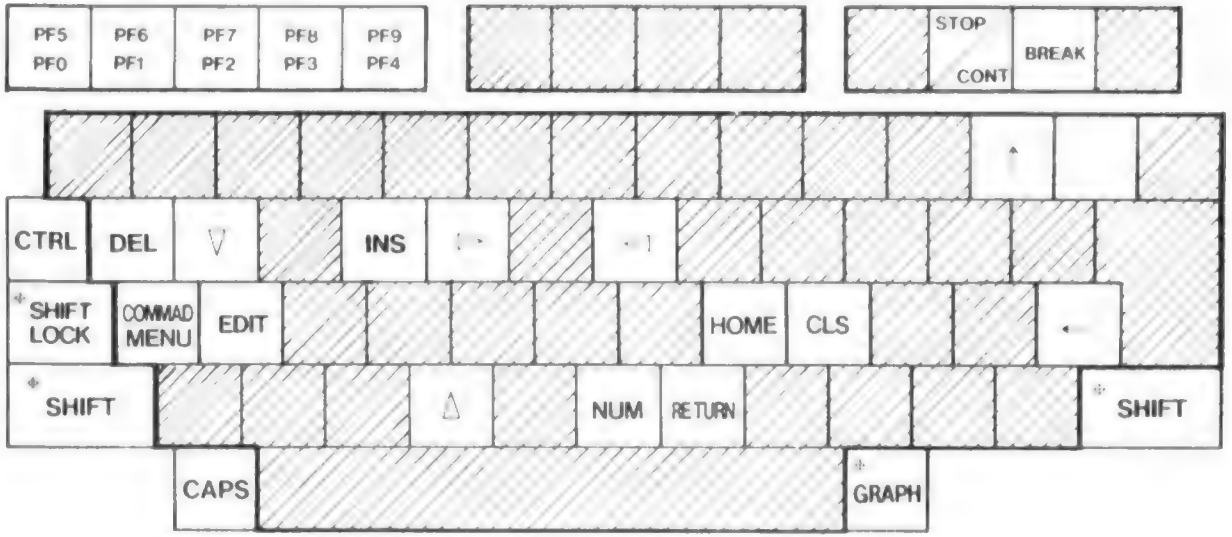
NUM-Tastenfunktionen



Tastenfunktionen der NUM-Umschalt-Betriebsart



Tastenfunktionen in der Control-Betriebsart



* Das gleiche gilt für den durch Kombination von CAPS, GRAPH und NUM erhaltenen Status.

Befehle, Anweisungen und Funktionen für C85-BASIC

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Systembefehle	Wahl eines Programmbereiches	PROG	–	PROG Programmbereichsnummer 1 Zeichen von 0 bis 9
	Einstellung (Rückstellung) eines Kennworts	PASS	–	PASS Kennwort Zeichenkonstante
	Löschen eines Programms	NEW	–	NEW[ALL]
	Anzeige des Status des Programmbereichs	SYSTEM	–	SYSTEM
	Definition des Inhalts einer PF-Taste	KEY	K.	KEY Funktionsnummer Tasteninhalt 1 Zeichen von 0 bis 9 Zeichenausdruck
	Definition des Inhalts einer PF-Taste	KEYLIST	K.L.	KEYLIST
	Löschen von Funktionen und durch den Benutzer definierten Funktionen. (Einstellen der Zeichenbereichsgröße und der BASIC-Obergrenze.)	CLEAR	–	CLEAR [Zeichenbereichsgröße, BASIC-Obergrenze] Numerischer Ausdruck, Numerischer Ausdruck * Die Zeichenbereichsgröße bei RESET ist 1.023 Bytes.
	Initialisieren des Hauptdirektzugriffsspeichers.	RESET	–	RESET
	Einstellung der CETL-Bereichsgröße.	AREA	–	AREA CETL-Bereichsgröße Numerischer Ausdruck
	Änderung von Zeilennummern	RENUM	–	RENUM [Neue Zeilennummer] Zeilennummer [, [Alte Zeilennummer]] [, [Zuwachs]] Zeilennummer Zeilennummer

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Systembefehle	Anzeige eines Programms auf der Flüssigkristallanzeige.	LIST	L.	LIST [Ausgangszeilennummer] [(-)] Zeilennummer [Endzeilennummer] Zeilennummer
	Anzeige der bezeichneten Zeile und initialisieren von Redigierbetriebsart	EDIT	ED.	EDIT [Zeilennummer] Zeilennummer
	Ausführung eines Programms	RUN	–	RUN [Ausführungsstartzeile] Zeilennummer
	Verfolgen der Ausführung eines Programms	TRON	–	TRON
	Ende des Verfolgens der Ausführung eines Programms	TROFF	–	TROFF
Ausführungssteuerung	Beendigung der Ausführung eines Programms	END	–	END
	Unterbrechung der Ausführung eines Programms	STOP	–	STOP
	Bedingungsloser Sprung	GOTO	G.	GOTO Zielzeilennummer Zeilennummer PROG Programmbereichsnummer Ein Zeichen von 0 bis 9
	Sprung zu einem Unterprogramm	GOSUB	GOS.	GOSUB Zielzeilennummer Zeilennummer PROG Programmbereichsnummer Ein Zeichen von 0 bis 9
	Rückkehr von einem Unterprogramm	RETURN	RET.	RETURN

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Ausführungssteuerung	Bedingter Sprung	IF ~ THEN ~ ELSE	– T. E.	IF Bedingung Numerischer Ausdruck THEN Anweisung [: Anweisung] * Ziel [ELSE { Anweisung [, Anweisung] * }] Ziel * Ziel { Zielzeilennummer / Zeilennummer / PROG Programmbereichsnummer } Ein Zeichen von 0 bis 9
	Bedingter Vielfachsprung	ON ~ GOTO	– G.	ON Bedingung GOTO Numerischer Ausdruck { Ziel [Ziel] [,[Ziel]] * , Ziel }
	Bedingter Unterprogrammabruf	ON ~ GOSUB	– GOS.	ON Bedingung Numerischer Ausdruck GOSUB { Ziel [Ziel] [, [Ziel]] * , Ziel } * Ziel { Zielzeilennummer / Zeilennummer / PROG Programmbereichsnummer } Ein Zeichen von 0 bis 9
	Wiederholung der Ausführung von Anweisungen für eine festgelegte Wiederholungsanzahl	FOR ~ NEXT	F. N.	FOR Steuervariablenname = Ausgangswert TO Endwert Numerischer Ausdruck (STEP Zuwachs] Numerischer Ausdruck NEXT [Steuervariablenname [, Steuervariablenname] *] * STEP kann als S abegekürzt werden.
Kommentar	Kommentaranweisung	REM	–	{ REM / ' } Kommentar Zeichenkette
Datenmanipulierung	Zuordnen eines Wertes zu einer Variablen	LET		[LET] { Numerischer Variablenname = Numerischer Ausdruck Zeichenvariablenname = Zeichenausdruck }
Lesen von Programmdaten	DATA-Anweisung	DATA	D.	DATA [Daten] [, [Daten]] * Konstante Konstante
	Lesen des Inhalts einer DATA-Anweisung	READ	–	READ Variablenname [, Variablenname] *

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Lesen von Programmdaten	Festlegung der Position der zu lesenden Daten	RESTORE	–	RESTORE [Zeilennummer / (Numerischer Ausdruck)]
LCD Bildschirmanzeige	Ausgabe von Daten zum Bildschirm LCD	PRINT	P.	PRINT [Ausgabeelement] { ; / . / ␣ } [Ausgabeelement] * * Ausgabeelemente sind Ausgabesteuerfunktionen, Zeichenausdrücke oder numerische Ausdrücke.
	Horizontale Bewegung des Positionsanzeigers zu einer bezeichneten Position	TAB F	–	TAB (Tabulatorspezifikation)
	Ausgabe von Daten in einem bezeichneten Format	PRINT USING	P.U.	PRINT USING Formatspezifikation / Zeichenausdruck ; Ausgabeelement { ; / , } Ausgabeelement]* [; / ,]
	Auffinden der Position des Positionsanzeigers	LOCATE	–	LOCATE X-Koordinate, Y-Koordinate (Numerischer Ausdruck)
	Löschen des Bildschirms	CLS F	–	CLS
Tastenfeldeingabe	Eingabe von Daten vom Tastenfeld	INPUT	I.	INPUT [Fragestellung (Mit einer Zeichenkonstanten beginnender Zeichenausdruck) ;] Variablenname [, Variablenname]*
	Holen eines Zeichens von einem Tastenpuffer	INKEY$ F	–	INKEY $
Variablengruppe	Festlegung des minimalen Wertes einer Variablengruppentiefstellung	OPTION BASE	–	OPTION BASE Minimaler Wert (Numerischer Ausdruck) * Der minimale Wert ist 0 oder 1.

Klassifizierung	Funktion		Reserviertes Wort	Abkürzung	Format
Variablengruppe	Erklärung einer Variablengruppe		DIM	–	DIM Variablengruppenname , (Maximaler Wert der Tiefstellung [, Maximaler Wert der Tiefstellung] *) [, Variablengruppenname (Maximaler Wert der Tiefstellung [, Maximaler Wert der Tiefstellung])]*
Maschinensprachenbezug	Abruf eines Maschinensprachenunterprogramms		CALL		CALL Adresse [,[argument]]*, argument / [, argument]
Typerklärung	Definition des Datentyps von Variablen und durch den Benutzer definierten Funktionen durch ein einziges Zeichen.	Reeller Typ einfacher Genauigkeit	DEF SNG	–	DEF SNG Zeichen [– Zeichen] [, Zeichen [, Zeichen [– Zeichen]]*
		Reeller Typ doppelter Genauigkeit	DEF DBL	–	DEF DBL Zeichen [– Zeichen] [, Zeichen [, Zeichen [– Zeichen]]*
		Zeichentyp	DEF STR F	–	DEF STR Zeichen [– Zeichen] [, Zeichen [, Zeichen [– Zeichen]]*
	Umwandlung numerischer Daten in einen bezeichneten Typ	Reeller Typ einfacher Genauigkeit	CSNG F	–	CSNG (Argument)
		Reeller Typ doppelter Genauigkeit	CDBL F		CDBL (Argument)

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Arithmetische Funktionen	Festlegung einer Winkelbetriebsart	ANGLE	–	ANGLE Winkelspezifikation Numerischer Ausdruck * 0: DEG 1: RAD 2: GRAD
	Trigonometrische Funktion ($\sin x$)	SIN F	–	SIN (Argument) * $-1440° <$ Argument $< 1440°$ Numerischer Ausdruck
	Trigonometrische Funktion ($\cos x$)	COS F	–	COS (Argument) * $-1440° <$ Argument $< 1440°$ Numerischer Ausdruck
	Trigonometrische Funktion ($\tan x$)	TAN F	–	TAN (Argument) * $-1440° <$ Argument $< 1440°$ Numerischer Ausdruck
	Inverse trigonometrische Funktion ($\sin^{-1} x$)	ASN F		ASN (Argument) $-90° \leqq$ ASN(X) $\leqq 90°$ Numerischer Ausdruck
	Inverse trigonometrische Funktion ($\cos^{-1} x$)	ACS F	–	ACS (Argument) $0° \leqq$ ACS(X) $\leqq 180°$ Numerischer Ausdruck
	Inverse trigonometrische Funktion ($\tan^{-1} x$)	ATN T	–	ATN (Argument) $-90° \leqq$ ATN(X) $\leqq 90°$ Numerischer Ausdruck
	Natürliche Logarithmusfunktion ($\log_e x$)	LOG F	–	LOG (Argument) * 0 < Argument Numerischer Ausdruck
	Gemeine Logarithmusfunktion ($\log_{10} x$)	LGT F	–	LGT (Argument) * 0 < Argument Numerischer Ausdruck
	Exponentialfunktion (e^x)	EXP F	–	EXP (Argument) * argument < 230 Numerischer Ausdruck
	Quadratwurzel ($\sqrt{x}$)	SQR F	–	SQR (Argument) * 0 < Argument Numerischer Ausdruck
	Absoluter Wert ($\|x\|$)	ABS F	–	ABS (Argument) Numerischer Ausdruck
	Annahme eines Wertes entsprechend dem Vorzeichen eines numerischen Wertes	SGN F	–	SGN (Argument) * 1 für negative Argumente, 0 für ein Argument von 0, 1 für positive Argumente. Numerischer Ausdruck

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Arithmetische Funktionen	Umwandlung eines gegebenen Wertes in eine ganze Zahl	INT F	–	INT (Argument) Numerischer Ausdruck
	Annahme des ganzzahligen Teils eines numerischen Wertes	FIX F	–	FIX (Argument) Numerischer Ausdruck
	Annahme des Bruchteils eines numerischen Wertes	FRAC F	–	FRAC (Argument) Numerischer Ausdruck
	Runden einer bezeichneten Stelle	ROUND F	–	ROUND (Argument , Stellenposition) Numerischer Ausdruck
	Erzeugen von Zufallszahlen	RND F	–	RND (Argument) Numerischer Ausdruck * Die gegenwärtige Zufallszahlenserie für ein Argument von 0, die nächste Zufallszahlenserie für negative Argumente.
	Änderung der Zufallszahlenserie	RANDOMIZE	–	RANDOMIZE
Zeichendatenmanipulierung	Annahme eines Zeichens eines festgelegten Kodes	CHR$ F	–	CHR$ (Kode) Numerischer Ausdruck * $0 \leq$ kode < 256
	Annahme des Kodes des ersten Zeichens	ASC F	–	ASC (Zeichenkette) Zeichenausdruck
	Umwandlung von Daten in eine Zeichenkette	STR$ F	–	STR$ (Argument) Numerischer Ausdruck
	Umwandlung von Daten in einen numerischen Wert	VAL F	–	VAL (Zeichenkette) Zeichenausdruck
	Ersetzen einer Zeichenkette	MID$	–	MID$ (Zeichenvariablenname, Position, Numerischer Ausdruck [, Anzahl von Zeichen]) = Numerischer Ausdruck Zeichenkette Zeichenausdruck

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Zeichendatenmanipulierung	Holen einer Zeichenkette von der linken Seite einer bezeichneten Zeichenkette	LEFT$ F		LEFT$ (Zeichenkette , Zeichenausdruck Anzahl von Zeichen) Numerischer Ausdruck
	Holen einer Zeichenkette von der rechten Seite einer bezeichneten Zeichenkette	RIGHT$ F	–	RIGHT$ (Zeichenkette , Zeichenausdruck Anzahl von Zeichen) Numerischer Ausdruck
	Festlegung der Länge einer Zeichenkette	LEN F	–	LEN (Zeichenkette) Zeichenausdruck
Funktionsdefinition	Definition einer durch den Benutzer definierten Funktion	DEFFN	–	DEFFN Funktionsname , [(Formalargument [, Formalargument]*)] = Ausdruck Einfacher Zeichenvariablenname
	Durch den Benutzer definierte Funktion	FN F	–	FN Funktionsname [(Realargument [, Realargument]*)] Ausdruck Ausdruck
Sonstige Funktionen	Bezeichnung der verbleibenden Speicherbytes	FRE F	–	FRE (Ausdruck) * Wert in Bytes.
	Annahme des Inhalts einer bezeichneten Speicheradresse	PEEK F	–	PEEK (Adresse) Numerischer Ausdruck * $0 \leqq$ adresse < 65536
	Schreiben von Daten in eine bezeichnete Speicheradresse	POKE	–	POKE Adresse, Daten Numerischer Ausdruck * $0 \leqq$ adresse < 65536 $0 \leqq$ daten < 256
	Angabe des Datums.	DATE$ F	–	DATE$
	Angabe der Zeit.	TIME$	–	TIME$
Statistische Verarbeitung	Initialisieren der statistischen Verarbeitungsfunktion	STAT CLEAR	–	STAT CLEAR

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Statistische Verabeitung	Eingabe von statistischen Daten	STAT	–	START X-Daten [, Y-Daten] (Numerischer Ausdruck)
	Zählen der Anzahl der statistischen Datenpunkte	CNT F	–	CNT
	Summe der X-Daten	SUMX F	–	SUMX
	Summe der Y-Daten	SUMY F	–	SUMY
	Summe der Quadrate der X-Daten	SUMX2 F	–	SUMX2
	Summe der Quadrate der Y-Daten	SUMY2 F	–	SUMY2
	Summe der Produkte der X- und Y-Daten	SUMXY F	–	SUMXY
	Mittelwert der X-Daten	MEANX F	–	MEANX
	Mittelwert der Y-Daten	MEANY F	–	MEANY
	Probenstandardabweichung der X-Daten	DSX F	–	SDX
	Probenstandardabweichung der Y-Daten	SDY F	–	SDY
	Standardabweichung von der Gesamtheit der X-Daten	SDXN F	–	SDXN
	Standardabweichung von der Gesamtheit der Y-Daten	SDYN F	–	SDYN
	Konstantenausdruck für lineare Regression	LRA F	–	LRA
	Linearer Regressionskoeffizient	LRB F	–	LRB

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Grafiksteuerung	Einstellen des Maßstabs einer Benutzerkoordinate	INIT	–	INIT (X-Koordinate , Y-Koordinate), Numerischer Ausdruck X-Zuwachs , Y-Zuwachs Numerischer Ausdruck
	Anzeige eines Punktes oder einer Geraden (Kontinuierliche Anzeige ist möglich).	DRAW DRAWC	– –	{ DRAW / DRAWC } [–] (X , Y) Numerischer Ausdruck [{ ' / – } (X, Y) Numerischer Ausdruck]* * DRAWC löscht einen Punkt oder eine Linie.
	Anzeige eines Rechtecks durch eine bezeichnete Funktion	QUAD QUADC	–	QUAD / QUADC [(X, Y)] – (X, Y) Numerischer Ausdruck / Numerischer Ausdruck * QUADC löscht ein Rechteck.
	Festlegung Punkt	POINT	–	POINT (X-Koordinate, Numerischer Ausdruck Y-Koordinate) Numerischer Ausdruck
Druckersteuerung	Drucken eines Programms	LLIST	LL.	LLIST [Startzeilennummer] Zeilennummer oder " . " [{ – / , } [Endzeilennummer]] Zeilennummer oder " . "
	Ausgabe von Zeichen zu einem Drucker	LPRINT	LP.	LPRINT [Ausgabeelement] [{ ; / , } [Ausgabeelement]] * * Ausgabeelement TAB (numerischer Ausdruck) Numerischer Ausdruck Zeichenausdruck
	Drucken von Leerstellen bis zu einer bezeichneten Position	TAB	–	TAB (Tabulatorspezifikation) Numerischer Ausdruck
	Drucken von Daten in einem bezeichneten Format	LPRINT USING	LP. U.	LPRING USING Formatspezifikation: Ausgabeelement [{ ; / , } Ausgabeelement] [{ ; / , }]

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Dateiverarbeitung	Festlegung der Anzahl der Dateipuffer	MOUNT	–	MOUNT Anzahl der Puffer Numerischer Ausdruck
	Erklärung des Beginns der Dateiverwendung	OPEN	–	OPEN Dateideskriptor [FOR { INPUT / OUTPUT } AS [#] Dateinummer
	Erklären des Endes der Dateiverwendung	CLOSE	–	CLOSE [[#] Dateinummer [, [#] Dateinummer] *
Sequentielle Verarbeitung	Ausgabe von Daten zu einer sequentiellen Datei	PRINT#	P. #	PRINT # Dateinummer [, Ausgabeelement [{ ; / , } [Ausgabeelement]] *]
	Ausgabe von Daten zu einer sequentiellen Datei in einem festgelegten Format	PRINT # ~ USING	P. # ~ U.	PRINT # Dateinummer , USING Numerischer Ausdruck Formatspezifikation ; Zeichenausdruck Ausgabeelement { , / ; } Numerischer Ausdruck oder Zeichenausdruck Ausgabeelement]* [{ , / ; }] Numerischer Ausdruck oder Zeichenausdruck
	Lesen von Daten von einer sequentiellen Datei	INPUT #	I. #	INPUT # Dateinummer , Numerischer Ausdruck Variablenname [, Variablenname] *
	Dateiende-Funktion	EOF	–	EOF (Dateinummer) Numerischer Ausdruck
Direktverarbeitung	Zuordnung von Zeichenvariablen zu Ein-/Ausgabepuffern	FIELD	–	FIELD [#] Dateinummer Variablenlänge Numerischer Ausdruck AS Zeichenvariablenname Zeichenvariablenname [, Variablenlänge Numerischer Ausdruck AS Zeichenvariablenname] * Einfacher Zeichenvariablenname

Klassifizierung	Funktion		Reserviertes Wort	Abkürzung	Format
Direktverarbeitung	Lesen eines Datensatzes in einen Ein-/Ausgabepuffer		GET	–	GET [#] Dateinummer Numerischer Ausdruck [, Datensatznummer] Numerischer Ausdruck
	Schreiben von Ein-/Ausgabepufferdaten in eine Datei		PUT	–	PUT [#] Dateinummer Numerischer Ausdruck [, Datensatznummer] Numerischer Ausdruck
	Eingabe von Daten rechtsausgerichtet in eine durch eine FIELD-Anweisung bezeichnete Variable.		RSET	–	RSET Zeichenvariablenname = Zeichenausdruck
	Eingabe von Daten linksausgerichtet in eine durch eine FIELD-Anweisung bezeichnete Variable		LSET	–	LSET Zeichenvariablenname = Zeichenausdruck
	Umwandlung eines numerischen Wertes in Zeichendaten durch festgelegte interne Darstellung	Reeller Typ einfacher Genauigkeit	MKS$ F	–	MKS$ (Argument) Numerischer Ausdruck
		Reeller Typ doppelter Genauigkeit	MKD$ F	–	MKD $ (Argument) Numerischer Ausdruck
	Umwandlung einer intern dargestellten Zeichenkette in einen numerischen Wert eines bezeichneten Typs	Reeller Typ einfacher Genauigkeit	CVS F		CVS (Argument) Zeichenausdruck
		Reeller Typ doppelter Genauigkeit	CVD F		CVD (Argument) Zeichenausdruck

Klassifizierung	Funktion	Reserviertes Wort	Abkürzung	Format
Direktverarbeitung	Festlegung des nächsten Datensatzes, der auf den letzten Datensatz der frei Direktverarbeitung folgt.	LOC	–	LOC (Dateinummer) Numerischer Ausdruck
	Festlegung der Anzahl der Datensätze einer Datei.	LOF	–	LOF (Dateinummer) Numerischer Ausdruck
Dateienhandhabung	Initialisieren des Mediums eines bezeichneten Gerätes.	FORMAT	–	FORMAT [Dateideskriptor] Zeichenausdruck
	Anzeige des Dateiattributs eines bezeichneten Gerätes.	FILES	–	FILES [Dateideskriptor] Zeichenausdruck
	Löschen einer bezeichneten Datei.	KILL	–	KILL [Dateideskriptor] Zeichenausdruck
Programmdatei	Einspeichern eines Programms.	SAVE	–	SAVE Dateideskriptor [, PASS WORD] Zeichenausdruck Zeichenausdruck [, A]
	Lesen eines Programms.	LOAD	–	LOAD Dateideskriptor [, R] Zeichenausdruck
Kassettentonbandgerät	Überprüfen einer Datei auf Kassettentonband.	VERIFY	–	VERIFY Dateideskriptor Zeichenausdruck

CETL-Funktionen

Klassifizierung	Funktion	Format
Befehl	Erstellen einer neuen Datei	N
	Einstellung der Reihenfolge für Dateneingabe	A
	Eingabe eines zusätzlichen Datensatzes oder Postens	I
	Löschen von Datensätzen oder Posten	D
	Bewegung von Datensätzen oder Posten	M
	Neubenennung einer Datei	R
	Löschen von Datensätzen	K
	Einschieben von Leerstellen in Datenzellen	B
	Sortieren innerhalb eines bezeichneten Punktes	S
	Durchführung von Aufsuchen nach Bedingung	F
	Sprung zu einer bezeichneten Datenzelle	J
	Drucken des Dateiinhalts auf dem Drucker	L
	Ausgabe von Dateidaten und Berechnungsergebnissen	T
	Berechnung des in einer Datei enthaltenen Berechnungsausdrucks	C
	Ausgabe von Daten zu einem externen Gerät	P
	Eingabe von Daten von einem externen Gerät	G
Verwaltungsfunktion	Rückkehr zur gegenwärtigen Datensatznummer	RC
	Rückkehr zur gegenwärtigen Postennummer	IT
	Rückgabe des Inhalts der Datenzelle im gegenwärtigen Punkt entsprechend dem bezeichneten Datensatz.	RC (Datensatzspezifikation) / Postennummer oder Datenwortname
	Rückgabe des Inhalts der Datenzelle im gegenwärtigen Datensatz entsprechend dem bezeichneten Posten.	IT (Postenspezifikation) / Datenwortnummer oder Datenwortname
	Lesen oder Schreiben des Inhalts der Datenzelle in der bezeichneten Datei entsprechend dem bezeichneten Datensatz und Posten.	FL ([Dateispezifikation] / Dateinummer oder Dateiname [Datensatzspezifikation] / Datensatznummer oder Datensatzname [Postenspezifikation]) / Postennummer oder Postenname

Klassifizierung	Funktion	Format
Verwaltungsfunktion	Berechnung der Summe der Daten im bezeichneten Datensatzbereich.	SUMRC (Datensatzspezifikation 1 , Datensatznummer oder Datensatzname Datensatzspezifikation 2) Datensatznummer oder Datensatzname
	Berechnung der Summe der Daten im bezeichneten Postenbereich.	SUMIT (Postenspezifikation 1 , Postennummer oder Postenname Postenspezifikation 2) Postennummer oder Postenname
Verknüpfungsbefehl	Abruf und Ausführung des BASIC-Programmbereichs von CETL.	PROG Programmbereichsnummer ; Ausdruck Eine Stelle von 0 bis 9

FEHLERKODETABELLE

Meldung	Bedeutung
OB-Fehler	Speicherüberlauf oder Systemstapelüberlauf.
SN-Fehler	Ein Befehl oder eine Anweisung hat einen Formatfehler.
IO-Fehler	Systeminterner Fehler während Datenübertragung zwischen Ein-/Ausgabevorrichtung und Computer.
ST-Fehler	Es steht kein Zeichenplatz mehr zur Verfügung, oder die Länge einer Zeichenkette überschreitet 255 Zeichen.
TC-Fehler	Ein Ausdruck ist zu komplex.
BV-Fehler	Überlauf des Ein-Ausgabespeichers.
NR-Fehler	Das Ein-/Ausgabegerät ist nicht bereit für Eingabe oder Ausgabe.
RW-Fehler	Ein Fehler im Ein-/Ausgabegerätbetrieb.
BN-Fehler	Die Dateinummer einer nicht geöffneten Datei wurde verwendet.
NF-Fehler	Ein bezeichneter Dateiname wurde nicht gefunden.
TS-Fehler	Eine falsche Spur-/Sektornummer wurde bezeichnet.
FL-Fehler	Es wurde versucht, Daten in nicht zur Verfügung stehenden Diskettenspeicherraum zu schreiben.
OV-Fehler	Das Ergebnis der Operation oder der Wert der Eingabedaten überschreitet den zulässigen Bereich.
MA-Fehler	Ein mathematischer Fehler wie z.B. Division durch 0.
DD-Fehler	Es wurde versucht, eine identische Variablengruppe oder durch den Benutzer definierte Funktion doppelt zu definieren.
BS-Fehler	Die Tiefstellung einer Variablengruppe überschreitet den zulässigen Bereich.
FC-Fehler	Falscher Abruf einer Funktion oder einer Anweisung.
UL-Fehler	Eine durch GOTO, GOSUB usw. bezeichnete Zeilennummer existiert nicht.
MO-Fehler	Ein erforderlicher Parameter ist in der Anweisung nicht definiert worden.
TM-Fehler	Die Variablentypen in einem Ausdruck (linke Seite, rechte Seite, Funktionsargument usw.) stimmen nicht überein.
PR-Fehler	Es wurde versucht, Daten in ein Programm oder eine Datei zu schreiben, die durch PASS, Schreibschutzkerbe usw. geschützt sind.
DATA-Fehler	Es wurde eine READ-Anweisung zum Lesen nicht existierender Daten ausgeführt.
FOR-Fehler	Einer NEXT-Anweisung ist keine FOR-Anweisung zugeordnet.

NEXT-Fehler	Einer FOR-Anweisung ist keine NEXT-Anweisung zugeordnet.
GOSUB-Fehler	Eine GOSUB-Anweisung ist nicht korrekt einer RETURN-Anweisung zugeordnet (es wurde eine RETURN-Anweisung gefunden, obwohl kein Abruf durch eine GOSUB-Anweisung erfolgte).
FN-Fehler	Die FN-Funktion ist nicht definiert.
FORMAT-Fehler	Ein Medium ist nicht formatiert worden, oder sein Format ist zerstört worden.
MOUNT-Fehler	Es wurden mehr externe Dateien geöffnet als Puffer zur Verfügung stehen.
FIELD-Fehler	Ein Bereich über 256 Bytes wurde in einer FIELD-Anweisung bezeichnet.
NAME-Fehler	Ein Dateiname oder ein Dateideskriptor wurde falsch festgelegt.
OPEN-Fehler	Es wurde versucht, Bezug auf eine nicht geöffnete Datei zu nehmen oder eine schon geöffnete Datei zu öffnen.
PASS-Fehler	Während PASS oder PASS ALL katalogisiert ist, wurde versucht, einen Befehl wie LIST auszuführen, oder ein falsches Kennwort wurde eingegeben.

(CETL)

Meldung	Bedeutung
OF-Fehler	Der CETL-Dateibereich reicht nicht aus.
OR-Fehler	Der Bereich des Arguments ist ungültig.
NE-Fehler	Der Datensatzname oder der Postenname existiert nicht.
IA-Fehler	Ungültige Antwort. Der gewählte Wert.

[A]

A	(CFTL)		X
ABS	(Funktion)	$\|x\|$	(5–10–4–2)
ACS	(Funktion)	$\cos^{-1}$	(5–10–1–3)
ANGLE	(Anweisung)	Festlegung der Winkeleinheit	(5–10–1–1)
ASC	(Funktion)		(5–11–2)
ASN	(Funktion)	$\sin^{-1}$	(5–10–1–3)
ATN	(Funktion)	$\tan^{-1}$	(5–10–1–3)

[B]

B	(CETL)		(13–7–8)

[C]

C	(CETL)		
CALL	(Anweisung)		(5–8–3)
CDBL	(Funktion)		(5–9–2)
CHRS	(Funktion)		(5–11–1)
CLEAR	(Befehl)		(4–1–7)
CLOSE	(Anweisung)		(9–2–3)
CLS	(Anweisung)		(5–5–6)
CNT	(Funktion)		(6–3)
COS	(Funktion)		(5–10–1–2)
CSNG	(Funktion)		(5–9–2)
CVD	(Funktion)		(9–4–7)
CVS	(Funktion)		(9–4–7)

[D]

D	(CETL)		(13–7–4)
DATA	(Anweisung)		(5–4–1)
DATES	(Funktion)		(5–13–4)
DEFFN	(Anweisung)		(5–12–1)
DEF SNG/DBL/STR	(Anweisung)		(5–9–1)
DIM	(Anweisung)		(5–7–2)
DRAW	(Anweisung)		(7–3–1)
DRAWC	(Anweisung)		(7–3–1)

[E]

EDIT	(Anweisung)		(4–2–3)
END	(Anweisung)		(5–1–1)
EOF	(Funktion)		(9–3–6)
EXP	(Funktion)	EXPe^x	(5–10–3–2)

[F]

F	(CETL)		(13–7–10)
FIELD [#]	(Befehl)		(9–4–1)
FILES	(Befehl)		(9–5–2)
FIX	(Funktion)		(5–10–4–5)
FN	(Funktion)		(5–12–2)
FORMAT	(Befehl)		(9–5–1)
FOR ~ NEXT	(Anweisung)		(5–1–8)
FRAC	(Funktion)		(5–10–4–6)
FRE	(Funktion)		(5–13–1)

[G]

G	(CETL)		(13–7–16)
GET [#]	(Anweisung)		(9–3–2)
GOTO	(Anweisung)		(5–1–3)
GOSUB/RETURN	(Anweisung)		(5–1–4–1/5–1–4–2)

[I]

I	(CETL)		(13–7–3)
IF ~ THEN ~ ELSE	(Anweisung)		(5–1–5)
INIT	(Anweisung)		(7–2–4)
INKEYS	(Funktion)		(5–6–3)
INPUT	(Anweisung)		(5–6–1)
INPUT#	(Anweisung)		(9–3–3)
INT	(Funktion)		(5–10–4–4)
IT	(CETL)		(13–8–2)
IT ( )	(CETL)		(13–8–4)

[J]

J	(CETL)		(13–7–11)

[K]

K	(CETL)		(13–7–7)
KEY	(Befehl)		(4–1–5)
KEY LIST	(Befehl)		(4–1–6)
KILL	(Befehl)		(9–5–3)

[L]

L	(CETL)		(13–7–12)
LEFT$	(Funktion)		(5–11–7)
LEN	(Funktion)		(5–11–10)
LET	(Anweisung)		(5–3–1)
LGT	(Funktion)	$\log_{10}X$	(5–10–3–1)
LIST	(Befehl)		(4–2–5)

LLIST	(Anweisung)		(8–1)
LOAD	(Befehl)		(9–6–2)
LOC	(Funktion)		(9–4–8)
LOCATE	(Anweisung)		(5–5–3)
LOF	(Funktion)		(9–4–9)
LOG	(Funktion)	$\log_e X$	(5–10–3–1)
LPRINT	(Anweisung)		(8–7)
LPRINT USING	(Anweisung)		(8–9)
LRA	(Funktion)		(6–3)
LRB	(Funktion)		(6–3)
LSET	(Anweisung)		(9–4–5)

[M]

M	(CETL)		(13–7–5)
MEANX	(Funktion)	x	(6–3)
MEANY	(Funktion)	y	(6–3)
MID$	(Funktion)		(5–11–5)
MKD$	(Funktion)		(9–4–6)
MKS$	(Funktion)		(9–4–6)
MOUNT	(Anweisung)		(9–2–1)

[N]

N	(CETL)		(13–7–1)
NEW	(Befehl)		(4–1–3)
NEW ALL	(Anweisung)		(4–1–3)

[O]

ON GOTO	(Anweisung)		(5–1–6)
ON GOSUB	(Anweisung)		(5–1–7)
OPEN	(Anweisung)		(9–2–2)
OPTION BASE	(Anweisung)		(5–7–1)

[P]

P	(CETL)		
PASS	(Befehl)		(4–1–2)
PEEK	(Funktion)		(5–13–2)
POINT	(Funktion)		(7–3–8)
POKE	(Anweisung)		(5–13–3)
PRINT	(Anweisung)		(5–5–1)
PRINT #	(Anweisung)		(9–3–1)
PRINT USING	(Anweisung)		(5–5–2)
PRINT #USING	(Anweisung)		(9–3–2)
PROG	(Befehl)		(4–1–1)
PUT [#]	(Anweisung)		(9–4–3)

[Q]

QUAD	(Anweisung)		(7–3)
QUADC	(Anweisung)		(7–3)

[R]

R	(CETL)		(13–7–6)
RC	(CETL)		(13–8–1)
RC ( )	(CETL)		(13–8–3)
READ	(Anweisung)		(5–4–2)
REM	(Anweisung)		(5–2)
RENUM	(Befehl)		(4–2–1)
RESET	(Anweisung)		(4–1–8)
RESTORE	(Anweisung)		(5–4–3)
RIGHT$	(Funktion)		(5–11–8)
RND	(Funktion)		(5–10–4–9)
ROUND	(Funktion)		(5–10–4–7)
RSET	(Anweisung)		(9–4–4)
RUN	(Befehl)		(4–3–1)

[S]

S	(CETL)		
SAVE	(Befehl)		(9–6–1)
SDX	(Funktion)		(6–3)
SDXN	(Funktion)		(6–3)
SDY	(Funktion)		(6–3)
SDYN	(Funktion)		(6–3)
SGN	(Funktion)		(5–10–4–3)
SIN	(Funktion)	sin	(5–10–1–2)
SQR	(Funktion)	$\sqrt{\ }\ x$	(5–10–4–1)
STAT	(Befehl)		(6–2)
STAT CLEAR	(Befehl)		(6–1)
STOP	(Anweisung)		(5–1–2)
STR$	(Funktion)		(5–11–3)
SUMIT	(CETL)		(13–8–7)
SUMRC	(CETL)		(13–8–6)
SUMX	(Funktion)	Σxy	(6–3)
SUMX2	(Funktion)	Σx^2	(6–3)
SUMXY	(Funktion)	Σxy	(6–3)
SUMY	(Funktion)	Σy	(6–3)
SUMY2	(Funktion)	Σy^2	(6–3)
SYSTEM	(Befehl)		(4–1–4)

[T]

T	(CETL)		(13–7–13)

TAB	(Funktion)		(5–5–1–2)
TAB (LPRINT)	(Funktion)		(8–8)
TAN	(Funktion)	tan	(5–10–1–2)
TROFF	(Befehl)		(4–3–4)
TRON	(Befehl)		(4–3–3)

[V]

VAL	(Funktion)		(5–11–4)
VERIFY	(Befehl)		(10–3)

	Verarbeitetes Zeichen	Groß- und Kleinbuchstaben, numerische Zeichen und Spezialsymbole		
Konstanten	Numerisches Zeichen	Einfache Genauigkeit	Anzahl der bedeutsamen Stellen	6
		Doppelte Genauigkeit	Anzahl der bedeutsamen Stellen	16
		Exponentenbereich	Maximum	+99
			Minimum	–99
	Zeichen	Max. Länge des Zeichenstrings		255
Variablen	Variablenname			Alphanumerische und Symbole 255
	Zeichen des Variablenattributs	Einfache Genauigkeit		!
		Doppelte Genauigkeit		#
		Zeichenstring		$
	Feldvariable	Max. Anzahl der Elemente		Abhängig von der Größe des Benutzerspeichers
		Anzahl der Dimensionen		Bis zu 3
Max. Dateinamen-Länge				8 + 3
Max. Länge der PF-Tastendefinition				15
Anweisung		Max. Länge der Zeichen pro Zeile		255
		Höchste Zeilennummer		64999
Arithmetische Operationen		Addition		+
		Subtraktion		–
		Multiplikation		*
		Division		/
		Exponentialrechnung		Λ
		Überschuß		MOD
Logische Operationen		Logisches AND		AND
		Logisches OR		OR
		Negation		NO
		Exklusives OR		XOR
Verschachtelung FOR-NEXT				(innerhalb der Speicherkapazität)
Unterprogramm-Verschachtelung				(innerhalb der Speicherkapazität)
Mehrfachanweisungszeilen				Möglich
Formatierte Ausgabe				USING-Anweisung